01 톰캣 5.x 설치와 환경 설정

부록 1절에서는 톰캣의 설치 과정과 서블릿 수행 환경 설정 그리고 교육용 웹 어플리케이션 디렉토리 생성과 등록 과정에 대하여 소개한다.

1.1 톰캣(Tomcat)이란?

톰캣은 Apache S/W Foundation에서 공개적으로 개발되고 배포되는 소프트웨어로서 전세계의 많은 개발자들에 의해 개발되고 있다.

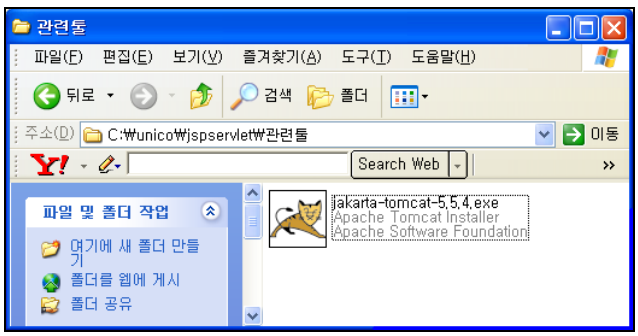
자바 서블릿 컨테이너 기능으로 출발한 톰캣은 현재 서블릿 컨테이너의 기능 외에도 웹 서버 기능과 JSP 컨테이너, 네이밍 서버 그리고 클러스터 등의 기능을 지원하는 WAS로서 사용될 수 있다. 톰캣은 자체적으로 보유하고 있는 내부 웹 서버와 함께 독립적으로 사용될 수도 있지만 아파치나 넷스케이프 엔터프라이즈 서버, IIS, 그리고 마이크로소프트의 PWS 등 다른 웹 서버와 함께 연동되어 사용될 수도 있다. 톰캣은 자바로 구현된 서버 기능의 어플리케이션이므로 자바 수행 환경을 필요로 한다.

1.2 톰캣의 설치 과정

JDK 1.4.2 또는 JDK 1.5을 먼저 설치한 다음에 톰캣 5.x를 설치하는데 여기에서는 톰캣 5.5.4를 설치하는 과정을 소개한다. 설치 전에 다음 사이트에서 톰캣 5.5.4를 다운로드 한다.

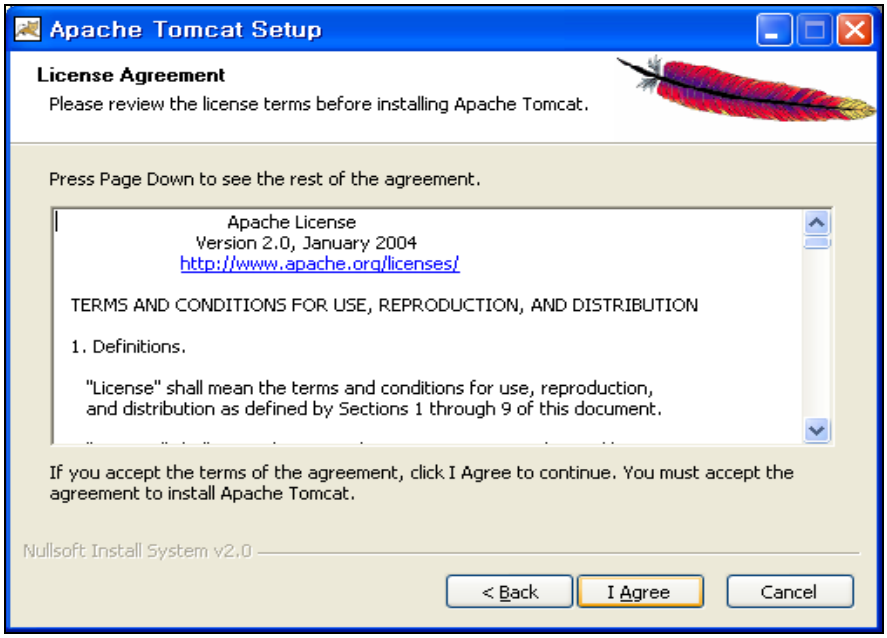
http://jakarta.apache.org/site/binindex.cgi#tomcat

다운로드된 jakarta-tomcat-5.5.4.exe를 더블클릭하여 설치과정을 시작한다.

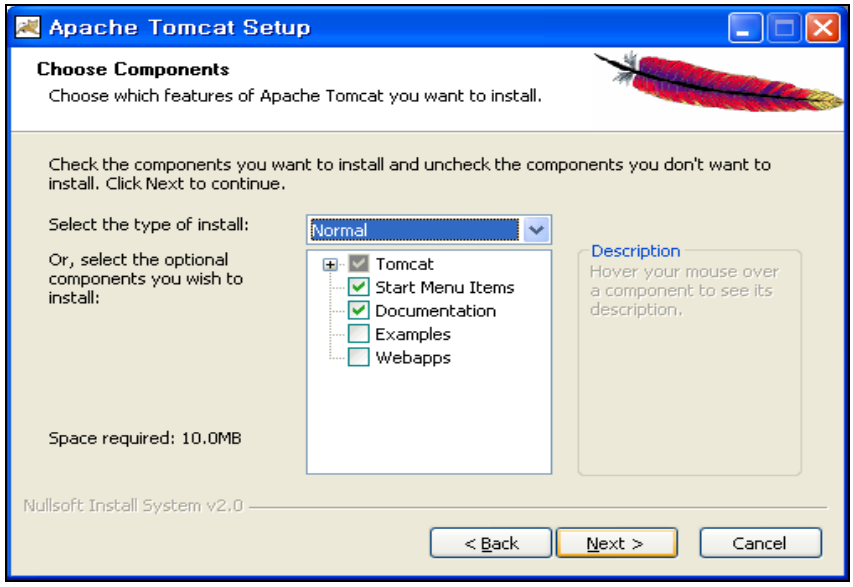


① 다음과 같이 Apache Tomcat Setup 화면이 출력되면 "Next" 버튼을 클릭한다.

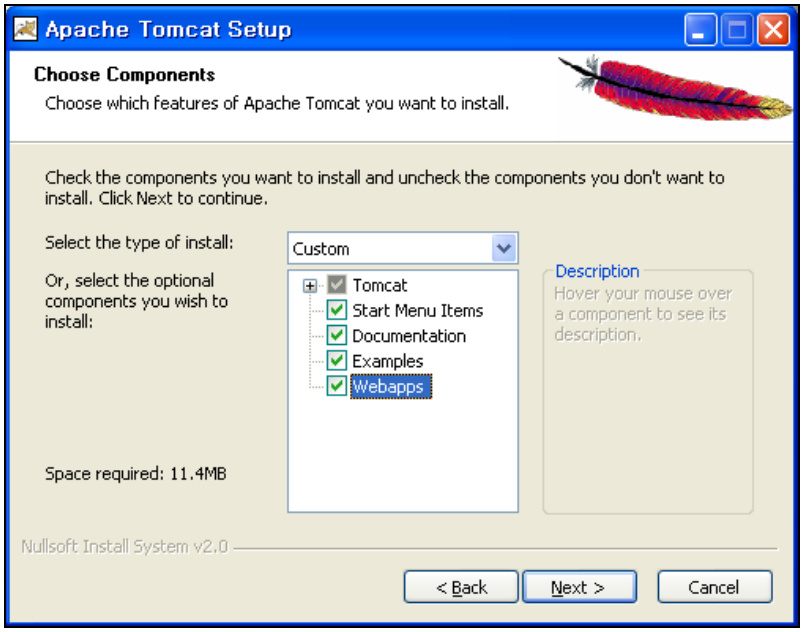
② 그러면 다음과 같이 License Agreement 화면이 출력된다. 설치를 계속하려면 "I Agree"를 선택한다.



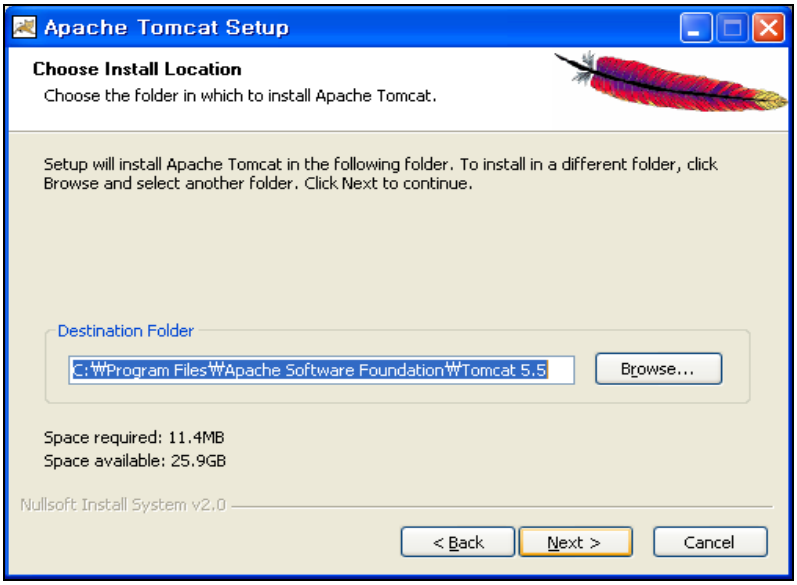
③ 다음과 같이 설치 컴포넌트를 선택하는 화면이 출력되면



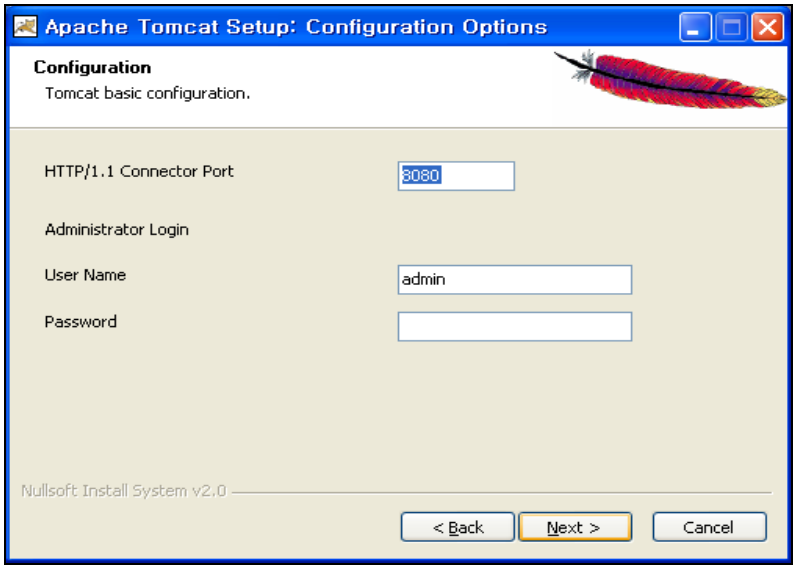
④ 다음과 같이 Examples 와 Webapps 도 선택한다. Examples는 톰캣이 내장하고 있는 서블릿과 JSP의 샘플 소스들이며 Webapps는 톰캣이 내장하고 있는 웹 어플리케이션 디렉토리들이다.



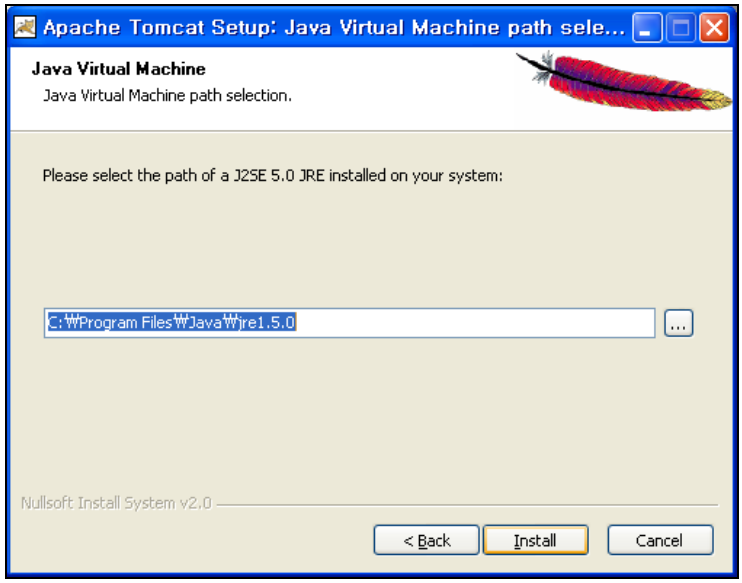
⑤ 다음과 같이 설치 디렉토리를 입력하는 화면이 출력된다. 특별한 경우가 아니면 정해진 디렉토리에 설치한다. 바로 "Next" 버튼을 클릭한다.



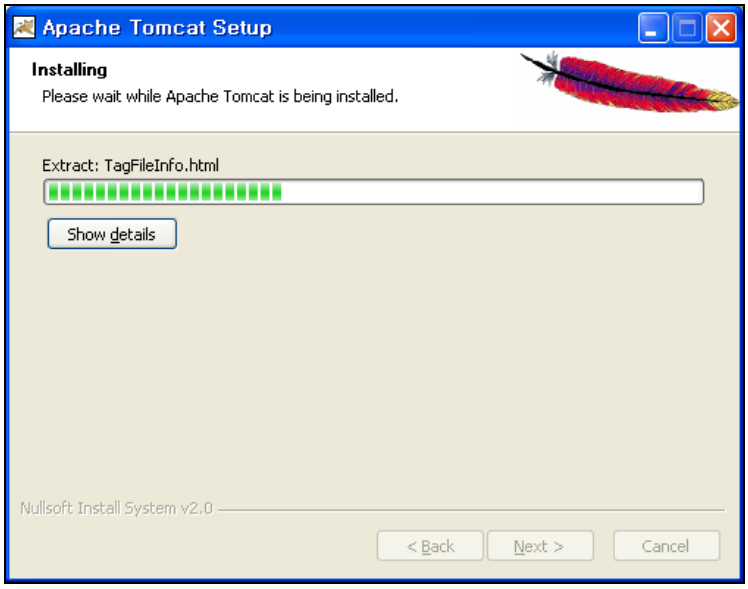
⑥ HTTP/1.1 Connector 포트 번호와 관리자 계정 그리고 관리자 패스워드를 입력하는 화면이 출력된다. 원하는 것을 입력해도 되지만 8080 포트 번호를 그대로 사용하고 관리자 계정도 admin을 그대로 사용한다. 패스워드만 입력한 후에 "Next" 버튼을 클릭한다.



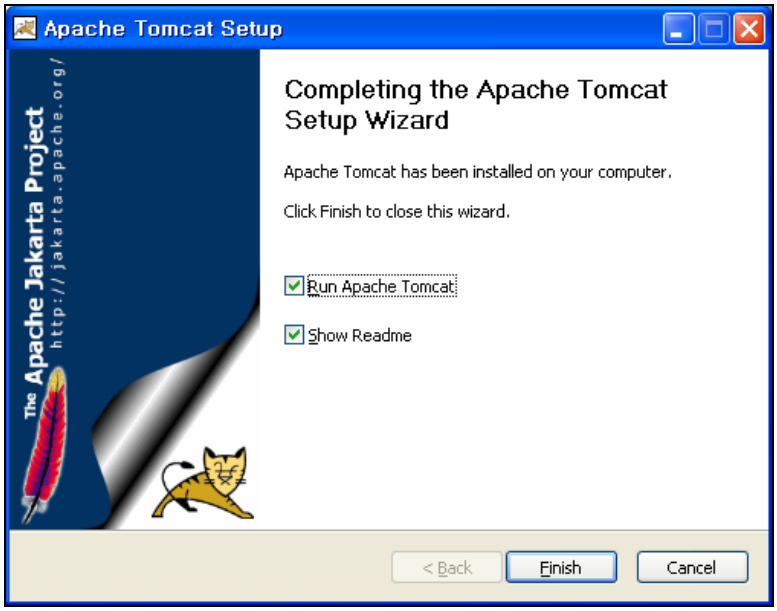
⑦ 다음과 같이 JVM의 패스를 설정하는 화면이 출력된다. 미리 JDK를 설치해 놓았으므로 톰캣이 알아서 JSK가 설치된 디렉토리를 찾는다. 확인만 한 후에 "Install" 버튼을 클릭한다.



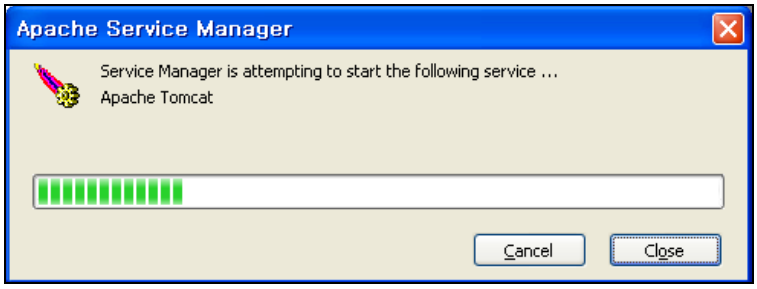
⑧ 다음 화면이 출력되면서 설치 과정이 진행된다.



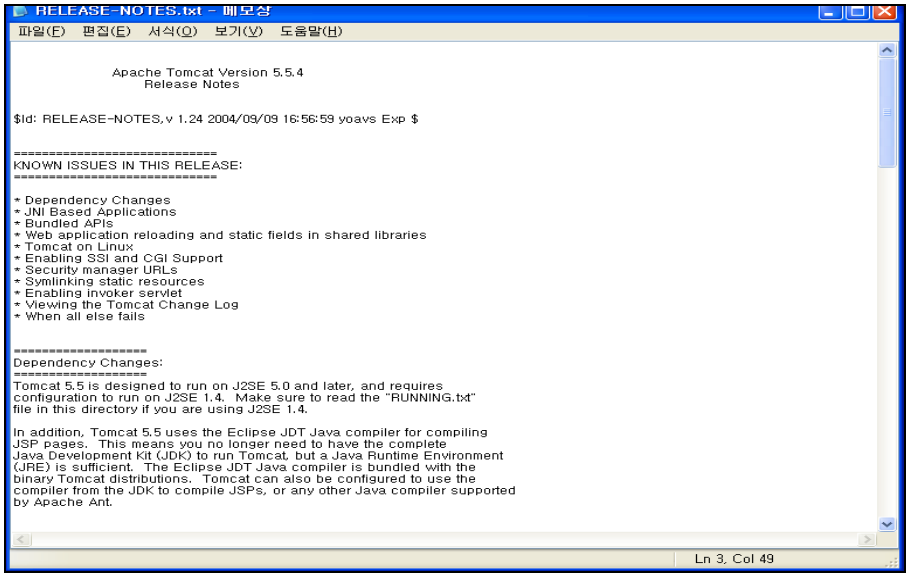
⑨ 설치가 모두 끝나면 다음 화면이 출력된다. "Finish" 버튼을 클릭한다.



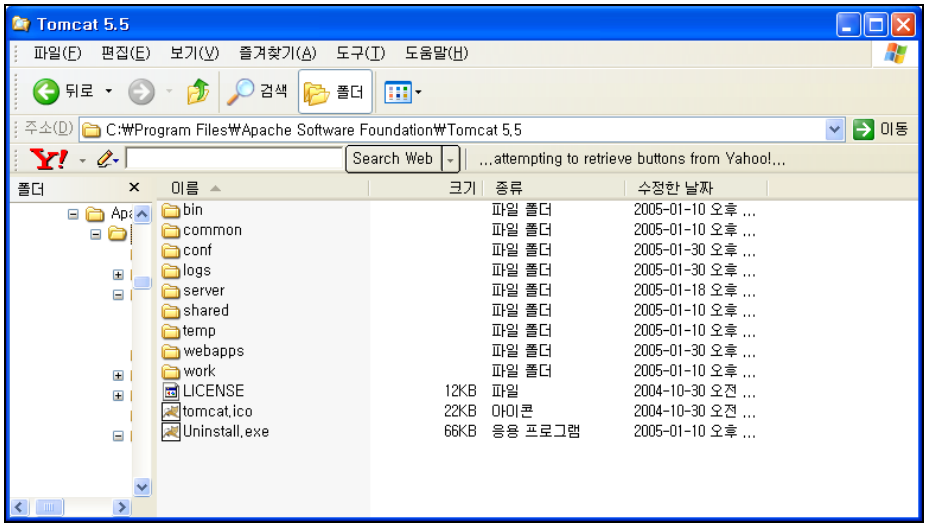
⑩ 다음과 같은 Apache Service Manager를 구성하는 윈도우가 출력된다.



⑪ 모두 진행한 후에 다음과 같이 톰캣의 RELEASE-NOTES.txt 파일의 내용이 출력된다.

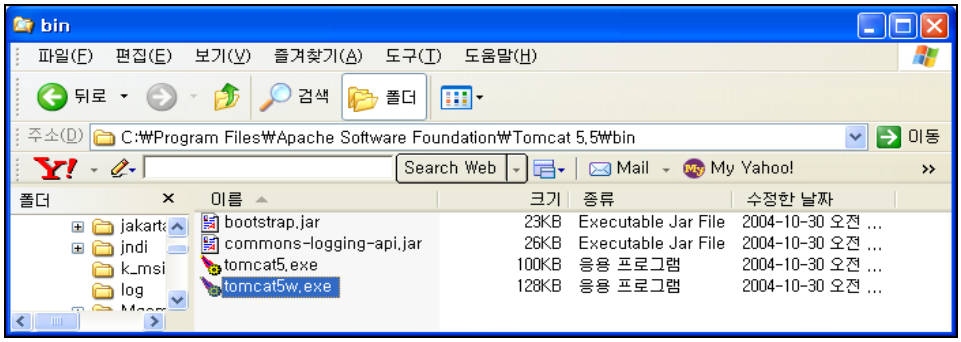


⑫ 설치 과정이 모두 끝났다. 제대로 설치가 진행되었는지 점검하기 위해 다음 화면과 같이 탐색기로 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5 디렉토리에 Tomcat 5.5가 설치된 것을 확인한다.

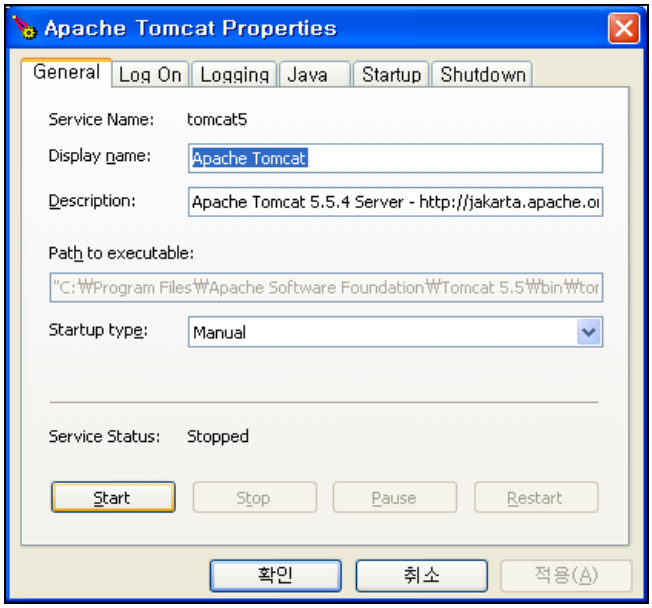


1.3 톰캣의 기동과 종료

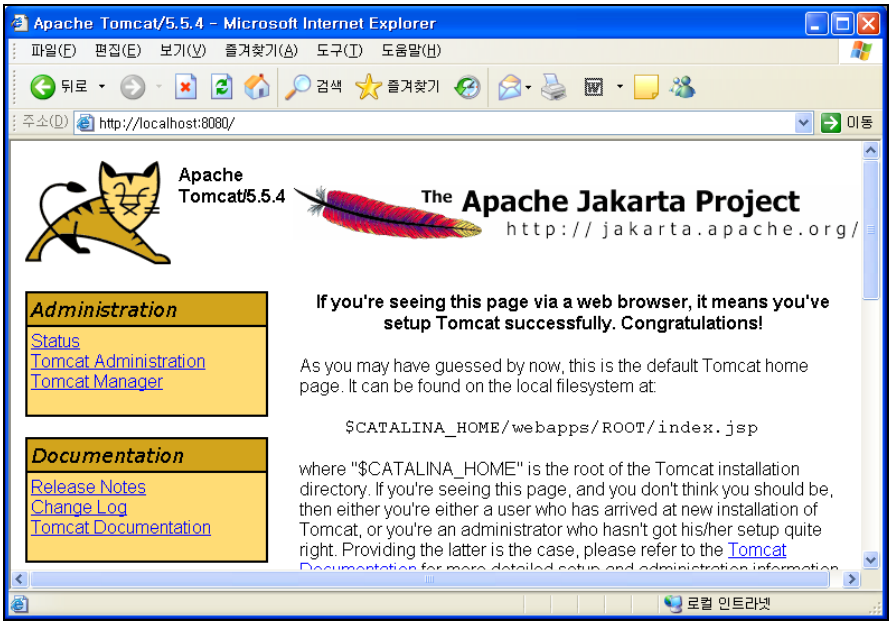
톰캣의 기동은 다음 화면과 같이 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩bin 디렉토리에서 tomcat5w.exe를 더블 클릭한다. 물론 tomcat5.exe를 수행시켜도 되지만 윈도우 버전을 수행시키는 것이 더 편하다.



다음 화면과 같이 Apache Tomcat Properties 윈도우가 출력되면 "start" 버튼을 클릭하여 기동하고 "stop" 버튼을 클릭하여 종료한다.



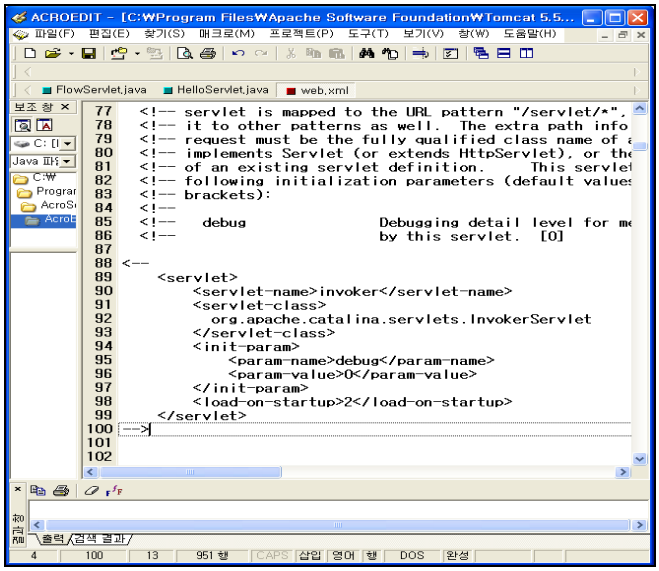
톰캣을 성공적으로 기동시킨 경우에는 브라우저에서 "http://localhost:8080/"으로 요청하여 다음과 같은 내용이 출력되는지 확인한다.



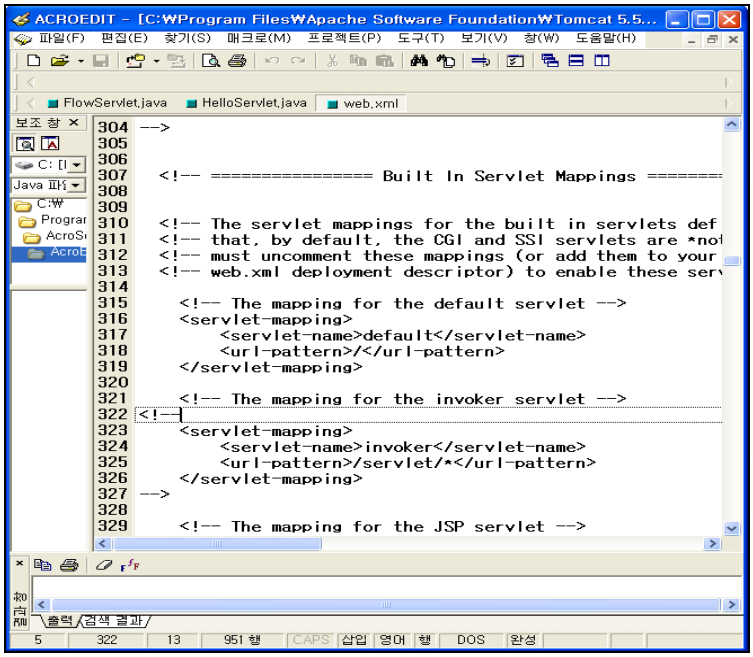
1.4 서블릿 수행 환경 설정

톰캣 5.x는 서블릿 수행에 대한 설정이 디폴트로 주석처리 되어 있다. 그러므로 서블릿을 테스트하려면 먼저 서블릿이 지원되는 환경을 설정해 주어야 한다. 설정 방법은 간단하다 주석처리 되어 있는 서블릿 수행 관련 환경설정 내용을 주석 해제해 주면 된다.

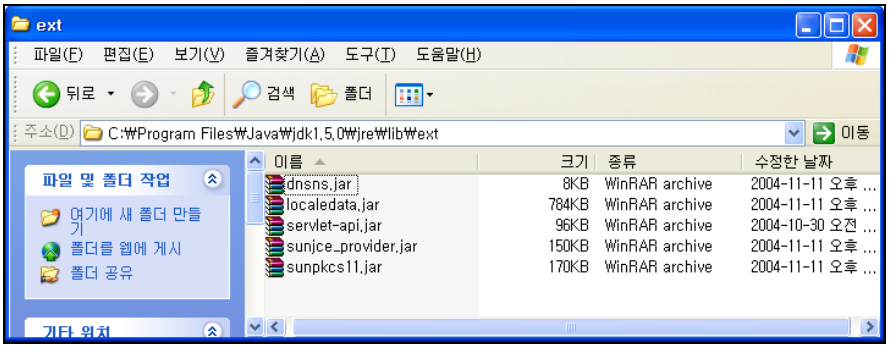
C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩conf₩web.xml을 에디터로 오픈하면 다음과 같이 InvokerServlet에 대한 환경 설정이 주석처리 되어 있는 것을 볼 수 있다. InvockerServlet이 인식될 수 있도록 주석을 해제한다.



그리고 클라이언트로부터 "/servlet/*"을 포함하는 파일 요청을 받으면 InvokerServlet이 수행되도록 다음 화면에 제시된 부분을 찾아 주석 해제한다.



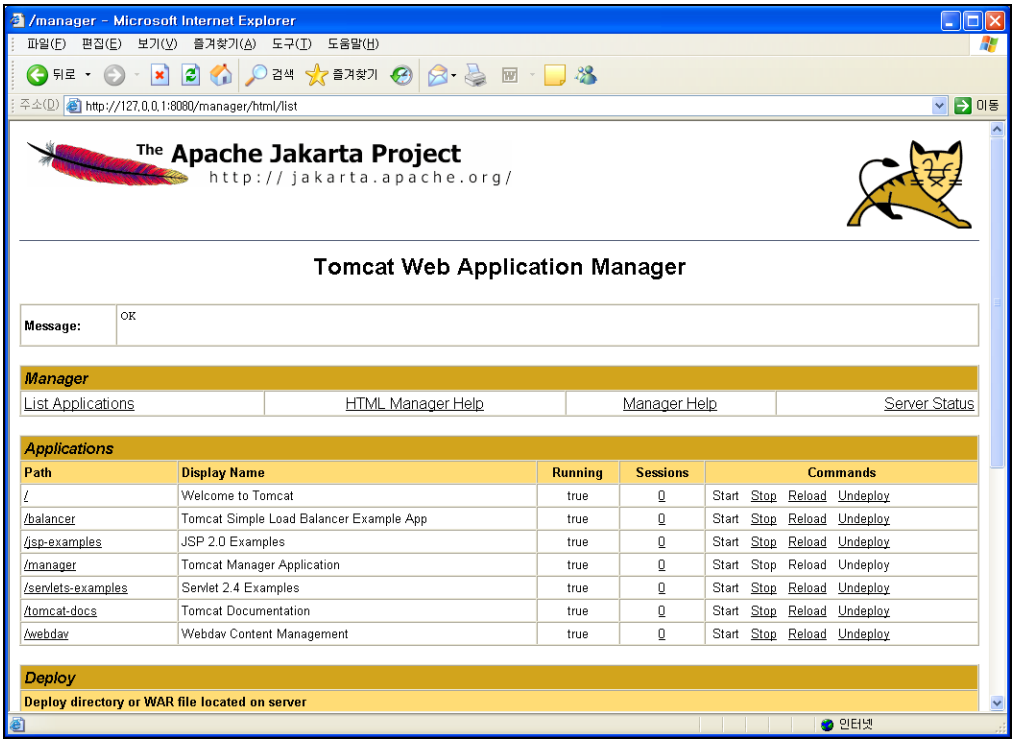
마지막으로 해주어야 할 작업은 Java 컴파일러가 서블릿 API를 인식할 수 있도록 다음과 같이 톰캣의 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩common₩lib 디렉토리에 저장되어 있는 서블릿 API의 압축 파일을 JDK가 설치된 디렉토리의 **jre₩lib₩ext** 디렉토리에 복사한다.



1.5 테스트용 웹 어플리케이션 등록

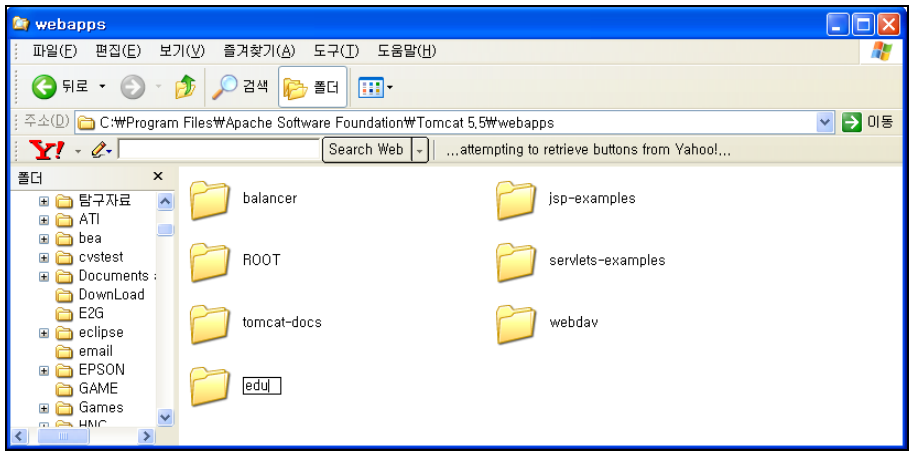
마지막으로 이 과정을 학습하는 동안 테스트할 예제들 즉, 서블릿, HTML 그리고 JSP 등을 저장해 놓고 테스트할 웹 어플리케이션 디렉토리를 톰캣에 등록하는 과정을 진행한다. 방법은 여러 가지가 있지만 여기에서는 Tomcat Manager를 사용하는 방법으로 소개한다. 윈도우의 시작메뉴를 선택하고 모든 프로그램 →

Apache Tomcat 5.5 → Tomcat Manager를 선택하면 다음과 같이 관리자 계정과 패스워드를 입력하라는 윈도우가 출력된다. 패스워드를 입력하고 "확인" 버튼을 클릭하면 다음과 같이

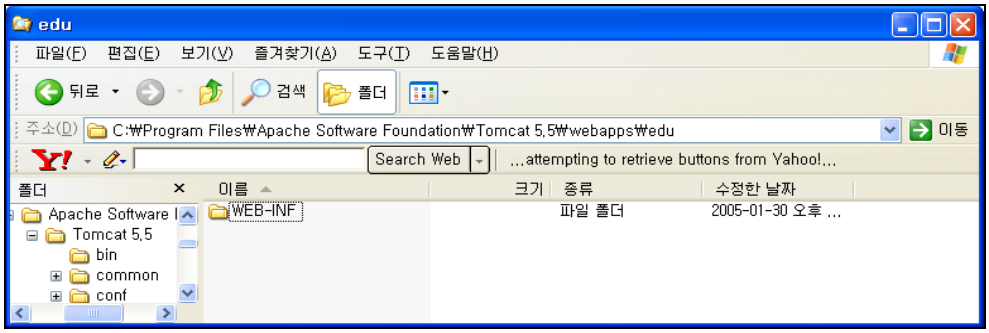


Tomcat Web Application Manager 화면이 브라우저에 출력된다. 이미 톰캣에는 여러 개의 웹 어플리케이션이 등록되어 있다는 것을 알 수 있다. Tomcat Manager 윈도우 화면은 점검만 하고 윈도우를 아이콘으로 만든 후에 다음 과정을 수행하여 웹 어플리케이션 디렉토리를 생성한다.

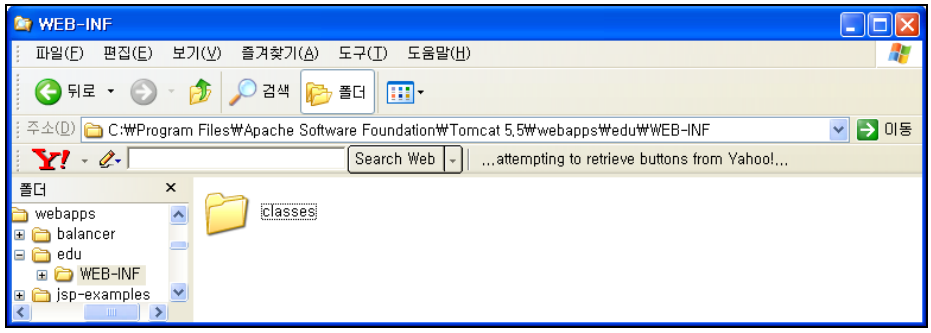
① C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩ 디렉토리에서 다음과 같이 "edu" 디렉토리를 생성한다.



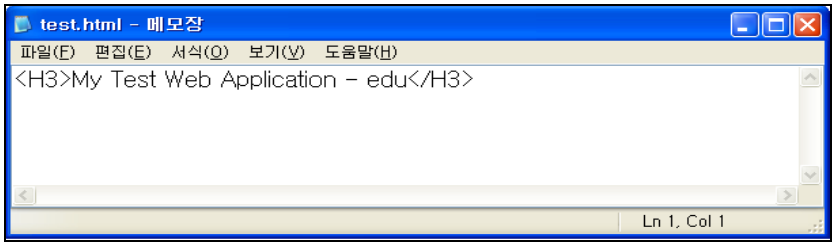
② C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu 디렉토리에서 "WEB-INF" 디렉토리를 생성한다.



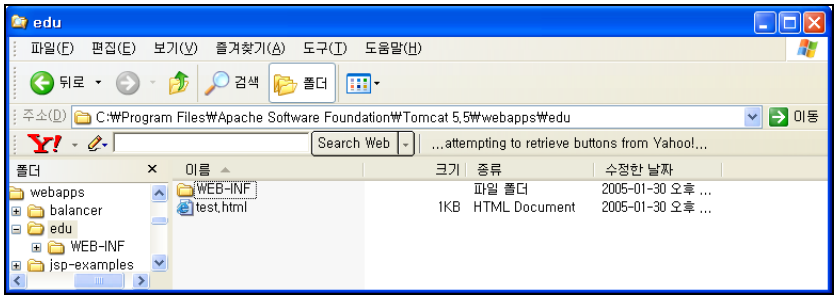
③ C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu ₩WEB-INF 디렉토리에서 classes 디렉토리를 생성한다.



④ 간단한 내용으로 test.html을 생성한다.



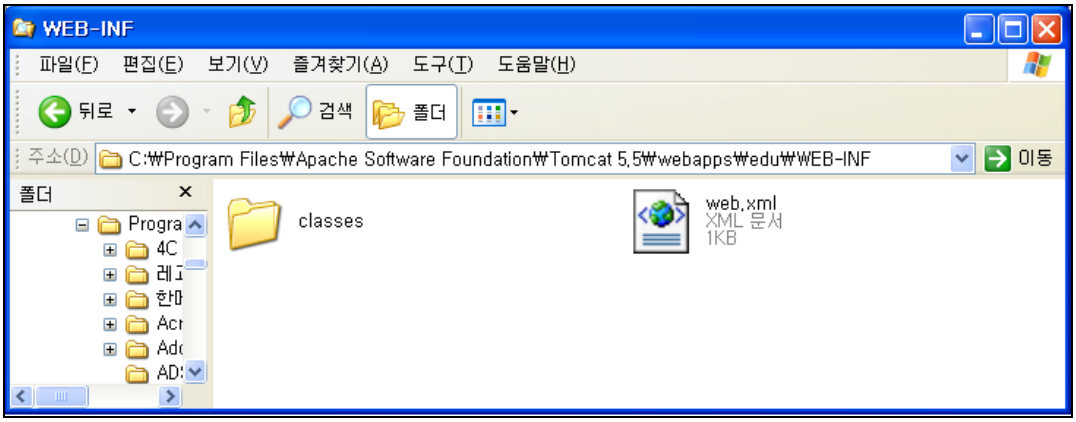
⑤ 작성된 test.html을 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu 디렉토리에 test.html이라는 파일명으로 저장한다.



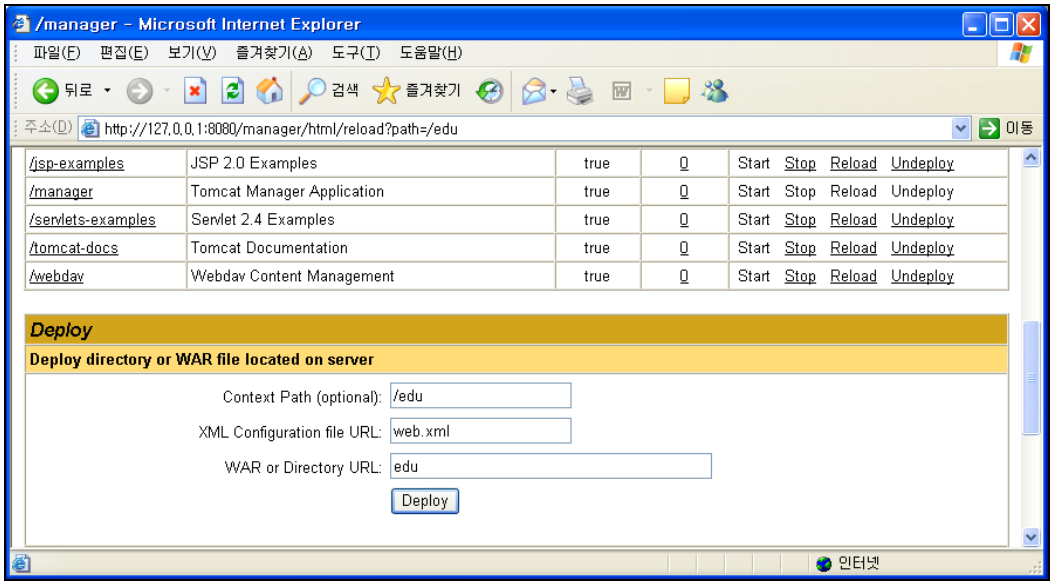
⑥ 다음과 같은 내용의 web.xml을 작성하여

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
    <description>TEST JSP Servlet</description>
    <display-name>EDUCATION TEST</display-name>
</web-app>
```

⑦ C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu ₩WEB-INF 디렉토리에 web.xml 이라는 파일명으로 저장한다.

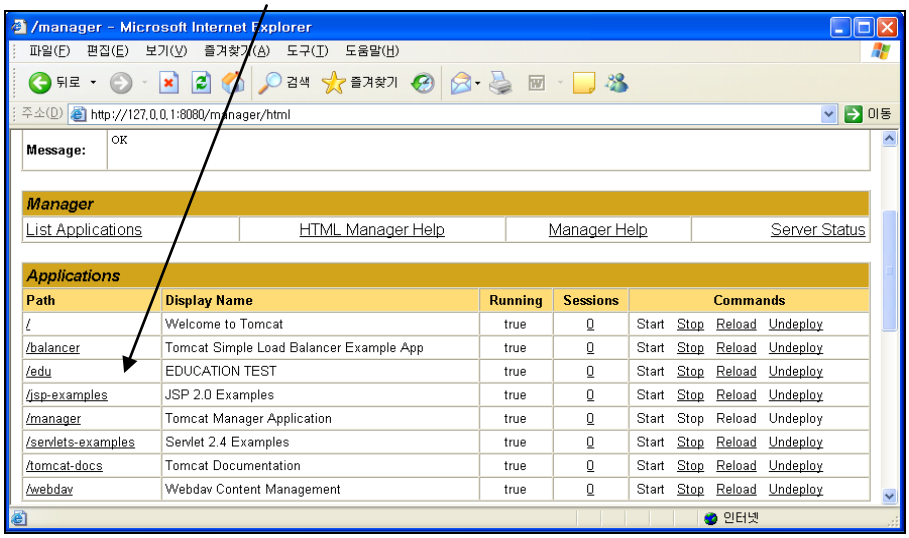


⑧ Tomcat Manager 윈도우 화면으로 가서 다음과 같이 Context Path에는 "/edu", XML Configuration file URL에는 "web.xml" 그리고 WAR or Directory URL에는 "edu"를 각각의 항목에 입력한 후에 "Deploy" 버튼을 클릭한다.



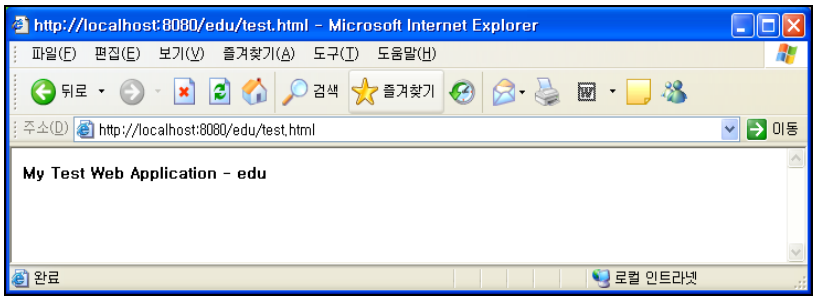
⑨ 그리고 나면 다음과 같이 Applications 테이블에 /edu 라는 패스를 갖는 새로운 Web Application 디렉토리가 생성된다. Display Name 열에 "EDUCATION TEST"라는 명칭이 출력되는 것을 볼 수 있다. 이 정보는 web.xml 에 작성된 다음 태그의 내용이 반영된 것이다.

```
<display-name>EDUCATION TEST</display-name>
```



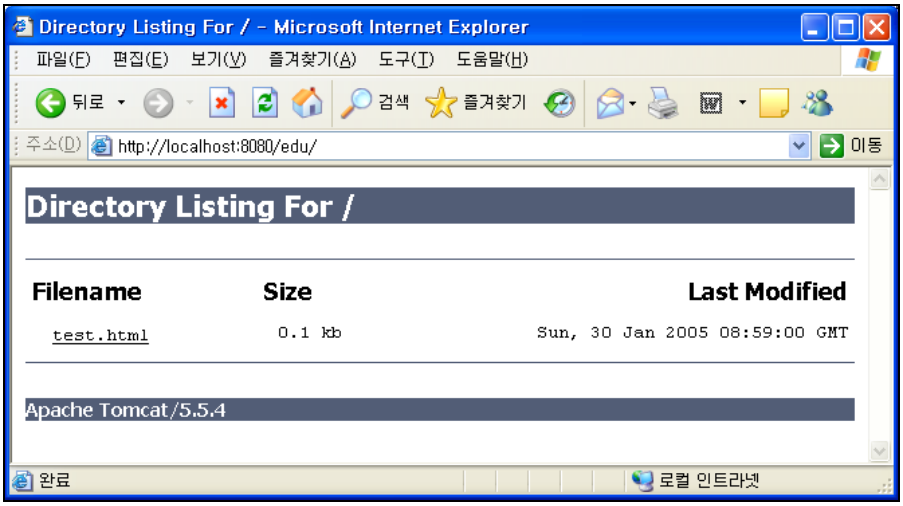
⑩ Running 열이 true인 것을 확인한 다음 다음과 같이 브라우저를 기동하여 "http://localhost:8080/edu/te- st.html"을 요청한다.

test.html 요청시 URI의 제일 앞에는 /edu 라는 Web Application을 구분하는 패스를 앞에 붙여 주면 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu 디렉토리에서 파일을 찾게 된다. 이 패스를 웹 어플리케이션 패스 또는 컨텍스트 패스라고 한다.



⑪ 이번에는 다음과 같이 파일명 없이 패스까지만 지정하여 요청해 본다. http://localhost:8080/edu

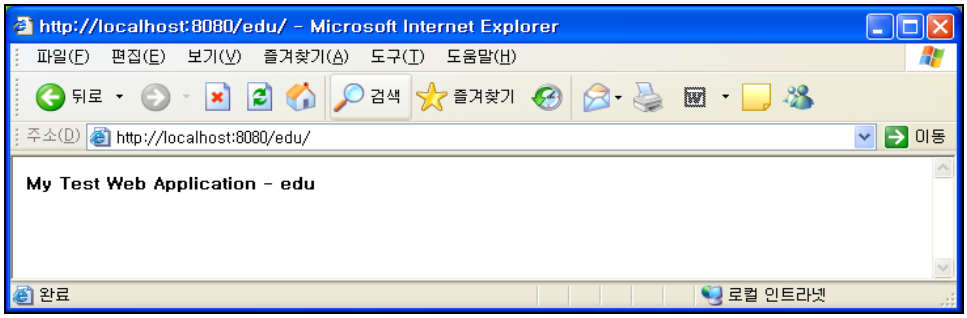
그러면 C:₩Program Files₩Apache Software Foundation₩Tomcat 5.5₩webapps₩edu디렉토리의 파일 리스트가 브라우저에 출력되는 것을 볼 수 있다.



⑫ web.xml 파일에서 <welcome-file-list>태그 항목을 추가한 후에 Tomcat Manager에서 패스가 "/edu" 인 Web Application 항목의 Commands 열에서 Reload 선택하여 새로운 web.xml 이 반영되도록 한다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
<description>TEST JSP Servlet</description>
<display-name>EDUCATION TEST</display-name>
<welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>test.html</welcome-file>
</welcome-file-list>
</web-app>
```

그러면 다음과 같이 파일명을 생략하고 정보를 요청하였을 때 <welcome-file> 엘리먼트에 정의된 순서대로 파일을 찾아 클라이언트로 응답한다.



02 Web.xml의 주요 엘리먼트

부록 2 절에서는 web.xml의 기능과 주요 엘리먼트들의 기능에 대하여 간단하게 소개한다.

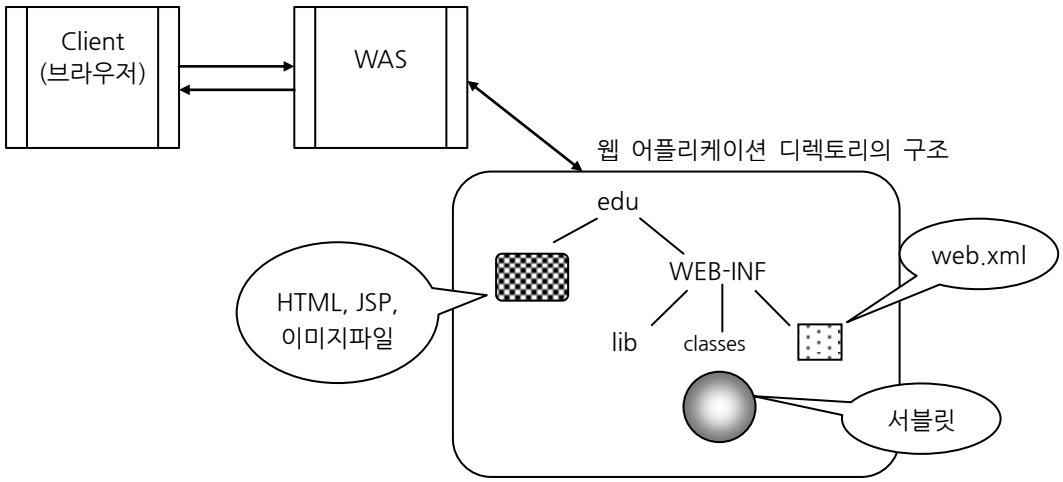
2.1 web.xml의 기능

1) **web.xml 이란?**

web.xml 은 웹 어플리케이션의 배치 기술(Deployment Descriptor)파일로서 XML 형식의 파일이다. web.xml 에는 서블릿 컨테이너가 서블릿과 JSP를 수행시킬 때 적용할 수 있는 다양한 환경 설정 내용을 작성한다. 다음은 web.xml에 작성하게 되는 주요 설정 내용이다.

ServletContext 의 초기 파라미터 세션의 유효시간 설정, 서블릿/JSP 에 대한 명칭 정의 서블릿/JSP 매핑, Mime Type 매핑, Welcom File list(기본 파일) 오류 페이지 설정, 리스너/필터 등록, 보안 설정

웹 어플리케이션은 임의의 디렉토리로 구성되며 JSP, HTML 기타 이미지 파일 등이 존재하는 디렉토리와 배치 기술자(DD파일, 환경 파일)파일이 존재하는 디렉토리 그리고 서블릿과 같은 서버상에서 수행되는 클래스 파일들이 존재하게 되는 디렉토리로 구성된다. 최상위 디렉토리(docbase)를 edu 디렉토리로 한다면 다음과 같은 구조가 된다.



2.2 web.xml의 주요 엘리먼트

web.xml은 <web-app> 태그로 시작하고 종료하는 문서로서 web.xml이 정의된 웹 어플리케이션의 동작과 관련된 다양한 환경 정보를 태그기반으로 정의하는 파일이다.

web.xml의 작성 규칙은 XML Schema로 정의되어 있다. 그러므로 XML Schema에 대한 사전 지식 없는 경우에는 작성 규칙을 보면서 이해한다는 것이 좀 어려운 것은 사실이다. web.xml 을 구성하는 모든 태그들에 대하여 한번에 학습하는 것은 도움이 되지 않는다. 서블릿을 학습하면서 web.xml에 작성되는 엘리먼트와 관련된 기술을 학습할 때 같이 점검해 보는 것이 좋은 방법이다.

서블릿 2.3까지의 web.xml의 자성 규칙은 DTD 파일이었고 2.4부터는 XML Schema 파일로 바뀌었다. 그러므로 다음과 같이 web.xml의 루트 엘리먼트인 <web-app> 엘리먼트에, 적용될 XML Schema를 선언해 주어야 한다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
    app_2_4.xsd" version="2.4">
```

web.xml에는 목적과 용도에 따라 필요한 태그만을 작성하면 되지만 태그의 작성 위치가 중요하며 XML문서이므로 대소문자 구분, 속성에 값 할당할 때 인용 부호 지정, 시작 태그와 종료 태그의 매핑에 대하여 주의 깊게 작성하여야 한다.

썬 사이트에서 "http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"을 요청하면 web.xml의 작성 규칙인 XML Schema 내용을 직접 점검해 볼 수 있다.

다음은 <web-app> 태그에 정의 가능한 주요 서브 태그들에 대한 리스트이다. 서브 태그는 XML Schema

에 정의된 순서대로 작성해야 한다.

<xsd:element name="description" type="j2ee:descriptionType" minOccurs="0"
maxOccurs="unbounded" />
  <xsd:element name="display-name" type="j2ee:display-nameType" minOccurs="0"
maxOccurs="unbounded" />
<xsd:element name="context-param" type="j2ee:param-valueType">
</xsd:element>
<xsd:element name="filter" type="j2ee:filterType" />
<xsd:element name="filter-mapping" type="j2ee:filter-mappingType" />
<xsd:element name="listener" type="j2ee:listenerType" />
<xsd:element name="servlet" type="j2ee:servletType" />
<xsd:element name="servlet-mapping" type="j2ee:servlet-mappingType" />
<xsd:element name="session-config" type="j2ee:session-configType" />
<xsd:element name="mime-mapping" type="j2ee:mime-mappingType" />
<xsd:element name="welcome-file-list" type="j2ee:welcome-file-listType" />
<xsd:element name="error-page" type="j2ee:error-pageType" />
<xsd:element name="jsp-config" type="j2ee:jsp-configType" />
<xsd:element name="security-constraint" type="j2ee:security-constraintType" />
<xsd:element name="login-config" type="j2ee:login-configType" />
<xsd:element name="security-role" type="j2ee:security-roleType" />

다음은 web.xml에 정의될 수 있는 주요 엘리먼트들에 대한 설명이다. 다음에 소개하는 엘리먼트들은 모두 <web-app>의 바로 아래 레벨의 태그들이다.

① <xsd:element name="description" type="j2ee:descriptionType" minOccurs="0" maxOccurs="unbounded" />
엘리먼트의 설명에 대해서 기술할 수 있는 태그이다. web-app에서 사용되는 경우에는 웹 어플리케이션의 설명 내용이 된다.

② <xsd:element name="display-name" type="j2ee:display-nameType" minOccurs="0"
maxOccurs="unbounded" />
<web-app> 태그에서 사용되면 웹 어플리케이션에 대한 이름을 지정할 수 있으며 filter, securityconstraint, servlet, web-app 안에서도 사용될 수 있다.

③ <xsd:element name="context-param" type="j2ee:param-valueType">
ServletContext의 초기값을 지정할 수 있으며 <web-app> 안에서 사용된다.
④ <xsd:element name="filter" type="j2ee:filterType" />
<xsd:element name="filter-mapping" type="j2ee:filter-mappingType" />

필터를 지정할 때 사용되는 태그이다. 필터는 웹 어플리케이션에 존재하는 모든 파일에 대하여 또는 원하는 서블릿과 JSP 단위로 설정하는 것이 가능하다.

⑤ <xsd:element name="listener" type="j2ee:listenerType" />
리스너 지정시 사용되는 태그이다. HttpServletRequest 객체, HttpSession 객체, ServletContext 객체에 대하여 리스너 정의가 가능하다.

⑥ <xsd:element name="servlet" type="j2ee:servletType" />
서블릿에 별칭을 정의하거나 초기 파라미터 등을 지정할 때 사용된다.

⑦ <xsd:element name="servlet-mapping" type="j2ee:servlet-mappingType" />
정해진 URI 패턴으로 요청하였을 때 어떠한 서블릿이 수행되도록 할 것인지 매핑하는 태그이다.

⑧ <xsd:element name="session-config" type="j2ee:session-configType" />
생성된 HttpSession 객체의 유효 시간을 설정하는 태그로, HttpSession 객체가 생생된 후로 정해진 시간 동안 클라이언트로부터 요청이 오지 않으면 생성된 HttpSession 객체는 자동 삭제 되는데 이 때 이 정해진 시간을 설정하는 태그이다.

⑨ <xsd:element name="mime-mapping" type="j2ee:mime-mappingType" />
웹 어플리케이션에서 처리되는 파일들에 대하여 파일의 확장자에 매핑되는 MIME-TYPE 정보를 설정한다.

⑩ <xsd:element name="welcome-file-list" type="j2ee:welcome-file-listType" />
<welcome-file-list>는 클라이언트에서 파일명 없이 요청하였을 때 대신 리턴되는 기본 파일을 설정하는 기능을 지원한다. 여러 파일들이 지정되면 지정된 순서로 우선 순위가 적용된다. 디폴트는 index.html 이다.

⑪ <xsd:element name="error-page" type="j2ee:error-pageType" />
오류 발생 시의 오류 처리 방식을 정의하는 태그이다.