



프로젝트로 배우는 자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 09. 웹 애플리케이션 아키텍처

목차

1. 웹 애플리케이션
2. 컨테이너와 배포 개념의 이해
3. 아파치 톰캣 시스템 관리
4. 웹 애플리케이션 배포하기

학습목표

- 웹 애플리케이션의 개념을 이해한다.
- 배포 서술자를 이해한다.
- 웹 애플리케이션 패키징 방법을 익힌다.
- 톰캣 관리자 기능과 사용 방법을 익힌다.

1. 웹 애플리케이션이란?

- 웹 애플리케이션이란 웹 서비스를 제공하기 위해 만들어진 일종의 프로그램을 말한다.
- 윈도우에서는 주로 .exe나 .com 파일을 통해 프로그램을 실행하지만, 웹 애플리케이션은 웹 브라우저의 URL 요청에 따라 웹서버를 통해 실행된다. 즉 웹에는 딱히 실행 파일이라고 부를 만한 형태가없다고 볼 수 있음.
- 웹 서비스는 여러 HTML 파일로 구성되어 있고, HTML 파일에는 화면을 구성하는 이미지 파일이나 플래시 애니메이션, css 및 js 파일 등 추가적인 콘텐츠가 포함되어 있다. 따라서 웹 애플리케이션은 이들 자원을 모두 포함한 개념으로 볼 수 있다. 물론 JSP, ASP, PHP와 같은 프로그램 요소들도 웹 애플리케이션의 구성요소에 포함된다.
- 이러한 구조는 스마트폰 애플리케이션에서도 동일하다. 스마트폰 애플리케이션도 여러 리소스로 구성된 번들의 형태며, 안드로이드 혹은 iOS의 런타임 프레임워크에 의해 관리되고 실행된다.

2. 웹 애플리케이션 콘텐츠 구성

- 웹 애플리케이션은 여러 유형의 콘텐츠로 구성된다. 프로그램을 구성하는 자원이라는 의미에서 리소스라는 표현을 쓰기도 한다.



[그림 9-1] 웹 애플리케이션 콘텐츠 유형

■ 동적 콘텐츠

- 서블릿, JSP와 같은 동적 콘텐츠는 단순한 텍스트가 아닌 프로그램으로서, 서비스 제공자의 의도나 사용자 및 상황에 따라 다른 내용을 제공하기 위한 콘텐츠이다.
- 예를들어, 인터넷 검색 서비스의 경우 사용자마다 검색하는 단어가 다를 수 있고, 동일한 단어라 할지라도 오늘 검색하는 것과 내일 검색 하는 것의 결과가 다를 수 있다.

■ 정적 콘텐츠

- HTML이 정적 콘텐츠의 대표적인 유형이다. HTML 파일에 작성된 내용은 파일 내용을 수정하기 전까지는 모든 사용자들에게 동일한 내용을 보여준다.

■ 보조 콘텐츠

- 이 밖에도 웹 애플리케이션 콘텐츠는 정적 콘텐츠에 부분적으로 동적인 서비스를 제공하기 위한 스타일시트나 자바스크립트 등과 같은 보조 콘텐츠도 포함될 수 있다.

3. 웹 애플리케이션 디렉터리 구성

- 우리가 흔히 사용하는 윈도우 기반 애플리케이션은 디렉터리를 사용할 때 몇 가지 규칙을 따른다. 예를 들어, 프로그램의 실행 파일의 대부분은 [Program Files] 폴더에 설치되고, 프로그램에서 사용하는 몇몇 파일들은 [Windows] 폴더에, 또 다른 파일은 [Windows\System] 혹은 [System32] 폴더에 위치하기도 한다.
- 또한 프로그램에서 참고할 일부 정보는 '레지스트리'라고 불리는 공간에 기록되기도 하고, 일부는 .conf, .ini 등의 파일 형태로 관리되기도 한다. 이처럼 하나의 프로그램은 여러 파일(자원)로 구성되어 있고, 목적에 따라 설치되는 위치가 다르다.
- 마찬가지로 웹 애플리케이션도 애플리케이션을 구성하는 콘텐츠의 유형과 역할에 따라 기본적으로 정해진 디렉터리 규칙이 있다. 웹 애플리케이션에 대한 디렉터리 규칙은 Java EE 스펙을 따른다. 참고로 맥 OSX의 경우는 애플리케이션이 번들 형태의 패키지 구조로 되어 있어 여러 곳에 나뉘어 저장되지 않는다. 스마트폰 애플리케이션도 이와 유사하다.

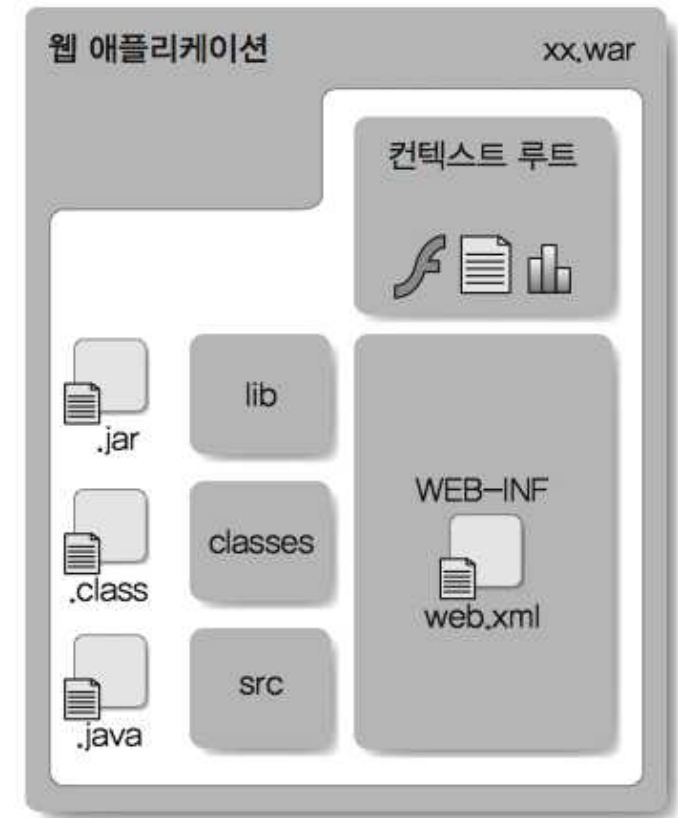
01. 웹 애플리케이션

■ 컨텍스트 루트(Context Root)

- 일명 홈 디렉터리(Home Directory), 혹은 도큐먼트 루트(Document Root)로 불리는 곳이다. 해당 애플리케이션의 메인 디렉터리가 된다. 대부분의 정적 콘텐츠가 위치하는 곳으로, 'WEB-INF' 하위 디렉터를 제외한 모든 하위 디렉터리의 콘텐츠가 서비스된다.

■ WEB-INF

- 'WEB-INF' 하위 디렉터리는 웹 애플리케이션의 배포 서술자인 web.xml 파일을 비롯해 애플리케이션에서 참조할 각종 XML, DTD, XSD, 그리고 태그 라이브러리 관련 파일 (TLD, 태그 파일) 등이 위치한다. 경우에 따라서 이들 파일들은 [WEB-INF] 폴더 아래에 별도의 폴더를 두어 관리하기도 한다.



[그림 9-2] 웹 애플리케이션 디렉터리 구조

■ WEB-INF/lib

- 애플리케이션에서 사용할 공통 라이브러리가 위치하는 곳이다. 보통 .jar 형태로 위치하며, JDBC 드라이버 등 서드 파티에서 제공하는 클래스 라이브러리를 참조하려고 사용하는 디렉터리다.

■ WEB-INF/classes

- 서블릿, 빈즈 클래스, 유틸리티 클래스의 컴파일된 클래스 파일이 위치하는 곳이다. 해당 소스의 패키지 구조에 따라 하위 폴더로 관리된다.

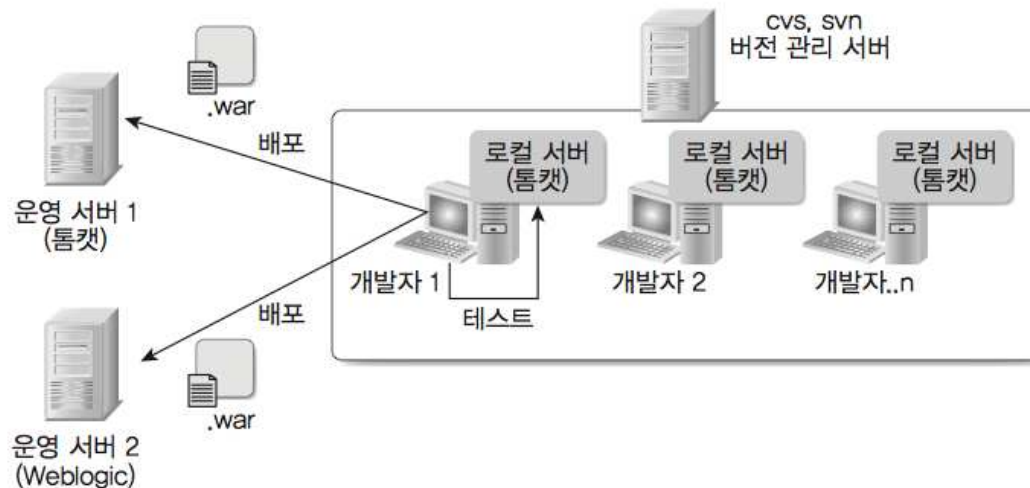
■ WEB-INF/src

- [src]의 경우 표준 웹 애플리케이션 디렉터리 구조라고 볼 수는 없으나, 빈즈 클래스나 서블릿 소스 파일을 관리하기 위한 디렉터리이다.

02. 컨테이너와 배포 개념의 이해

1. 컨테이너와 개발환경

- 컨테이너는 웹 애플리케이션을 실행하기 위한 환경으로, 사용자의 요청을 받아들이고 처리하는 역할을 수행하는 일종의 서버 프로그램을 말한다.
- 이론적으로 Java EE 표준 스펙을 준수할 경우, 동일한 애플리케이션은 서로 다른 업체의 컨테이너(IBM WAS, Oracle WAS, 톰캣 등)에서 어떤 수정도 없이 실행할 수 있다.
- 소규모 개발이나 학습을 위한 경우에는 대부분 동일할 것이고, 일정 규모 이상의 실제 서비스를 운영하는 경우에는 개발 컴퓨터와 실제 서비스가 운영되는 컴퓨터가 다를 것이다.
- 일반적으로 개발자의 컴퓨터에서 개발 및 테스트가 이루어지고 최종적인 버전이 완성되었을 때 실제 서비스 서버에 배포된다.
- 다음은 중소규모 이상의 웹 애플리케이션의 개발을 위한 환경이다.



[그림 9-3] 컨테이너와 웹 애플리케이션 배포

02. 컨테이너와 배포 개념의 이해

2. 웹 애플리케이션의 배포

- 작성된 웹 애플리케이션을 컨테이너에서 실행하려면 배포라는 작업이 필요하다. 배포는 쉽게 말해서, 윈도우에서 setup.exe 등을 이용해 사용할 프로그램을 설치하는 과정이라고 보면 된다.
- 각 컨테이너마다 배포 방법에는 다소 차이가 있을 수 있으며, 전용 배포 툴이나 웹 기반의 배포 프로그램을 제공하기도 한다.
- **톰캣 디렉터리에 직접 파일로 배포하는 방법**
 - 가장 간단하고 쉽게 할 수 있는 방법으로, 예전 톰캣에서도 동일한 방법으로 운영할 수 있다. 그러나 톰캣 디렉터리를 이용하는 것은 톰캣 버전 교체 혹은 톰캣 재설치 상황에서 배포한 애플리케이션이 지워질 수 있으므로 권장하지 않는다.
- **Tomcat Manager를 이용한 방법**
 - 톰캣에 기본적으로 포함되어 있는 웹 기반의 관리도구다. 비교적 쉽게 웹 애플리케이션을 배포하거나 관리할 수 있으며, 관리자에 대한 접근 권한 등을 설정해서 어느 정도의 보안 문제도 해결할 수 있다. 따라서 이 방법을 권장하는 편이다.
- **Tomcat Client Deployer를 이용하는 방법**
 - TCD(Tomcat Client Deployer)는 단순한 배포 툴이라기보다 유틸리티에 가까우며 톰캣에서 추가적인 서비스를 제공하고 Ant 기반의 톰캣 태스크를 정의해놓은 패키지다. 서버 관리자보다는 개발자들에게 유용한 툴로써, 간단한 명령으로 웹 애플리케이션을 배포하고 관리할 수 있다. Ant에 능숙한 개발자라면 이 방법도 괜찮지만 명령 행 방식이므로 사용이 다소 불편한 점도 있다.

02. 컨테이너와 배포 개념의 이해

3. 배포 서술자

- 배포 서술자(DD, Deployment Descriptor)는 Java EE 스펙으로 웹 애플리케이션의 기본적인 설정을 위해 작성하는 파일로, 보통은 WEB-INF/web.xml 을 말한다.

■ 배포서술자 사용 목적

- 컨테이너 호환성 유지
 - 웹 애플리케이션의 배포와 관련된 정보를 제공함으로써, 서로 다른 컨테이너에서도 별도의 설정 없이 동일한 애플리케이션을 운영할 수 있다.
- 효율적인 애플리케이션 유지보수
 - 특정 서블릿 혹은 전체 애플리케이션에서 공유하기 위한 초기화 매개변수를 설정할 수 있으므로 애플리케이션 프로그램 간에 데이터를 쉽게 공유할 수 있다.
- 유연한 애플리케이션 운영
 - 서블릿 매핑 등 다양한 정보를 텍스트 기반으로 설정할 수 있으므로, 서비스 운영 중 프로그램을 수정하지 않고도 애플리케이션의 동작 등을 조정할 수 있다.

02. 컨테이너와 배포 개념의 이해

4. web.xml 세부 구조

- web.xml 파일은 XML 형식으로 작성된 일반 텍스트 파일이다. 서블릿 스펙 2.4 이후부터는 XML 스키마 형식을 따르고 있으며 3.0 이상부터는 애너테이션 기반의 설정을 지원하므로 web.xml 사용이 예전보다는 줄어들게 되었다.
- web.xml 설정은 웹 애플리케이션 실행과 관련이 있고, 설정이 잘못될 경우 컨테이너에서 해당 웹 애플리케이션을 제대로 실행하지 못하는 문제가 발생할수도 있으니 설정에 주의한다.

[표 9-1] web-app_3_0.xsd 주요 구성요소 리스트

구성요소	설명
distributable	클러스터 같은 분산된 환경에서 사용할 것인지 설정 하는 부분이다.
context-param	서블릿 컨텍스트의 초기 매개변수를 설정한다. 설정된 내용은 모든 서블릿과 JSP에서 참조할 수 있다.
filter	필터 서블릿을 등록하는 부분이다.
filter-mapping	필터 서블릿의 URL 매핑을 지정하는 부분이다.
listener	리스너 서블릿을 등록하는 부분이다.

02. 컨테이너와 배포 개념의 이해

구성요소	설명
servlet	서블릿을 등록하는 부분이다.
servlet-mapping	서블릿의 URL 매핑을 지정하는 부분이다.
session-config	현재 애플리케이션에 대한 세션과 관련된 설정을 하는 부분이다.
mime-mapping	애플리케이션에서 처리할 파일 형태에 대한 매핑 부분이다.
welcome-file-list	index.html, index.jsp 등 초기 로딩 파일을 지정하는 부분이다.
error-page	서버 오류 코드에 따른 오류 페이지를 지정한다. 예를 들어 404 오류는 특정 파일을 찾을 수 없는 경우 발생하는 오류다. 해당 오류에 대해서 별도 파일로 오류 메시지를 처리할 수 있다.
jsp-config	여러 jsp에서 include 지시어를 사용해서 특정 페이지들을 포함할 경우 jsp-config 설정을 통해 이 부분을 자동화할 수 있다. taglib 지시어에도 활용할 수 있다.
security-constraint	애플리케이션 접근에 대한 보안 설정 부분이다.
login-config	웹 애플리케이션 접근에 대해 로그인 방식으로 인증하기 위한 설정이다. 사용자 설정은 tomcat-users.xml에서 해주어야 한다.

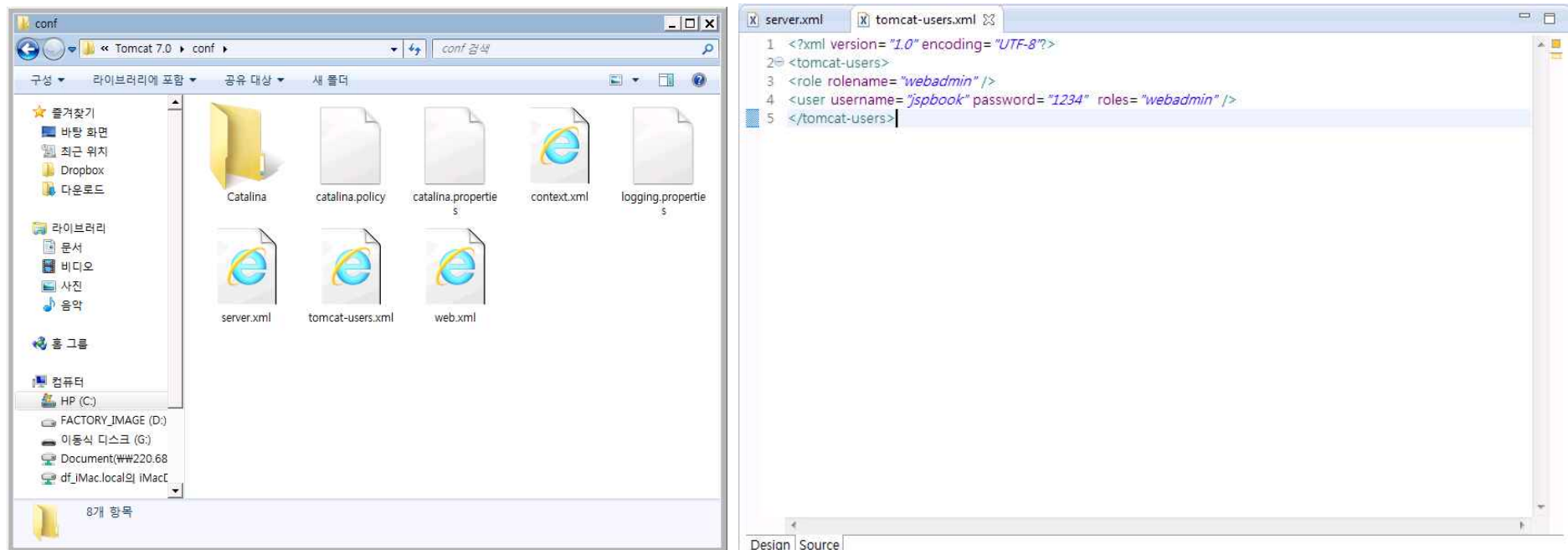
■ [예제] web.xml 파일의 구조(예시)

- 교재 p.403 ~ 404 참고

03. 아파치 톰캣 시스템 관리

1. 아파치 톰캣 사용자 설정

- 서버 시스템은 항상 보안과 밀접한 관련이 있다. 네트워크나 운영체제 보안 이외에도 웹 애플리케이션에 대한 보안 설정이 필요하다. 가장 기본이 되는 것은 사용자를 등록하고 보안이 요구되는 웹 리소스 접근에 대해 권한 관리를 하는 것이다.
- 톰캣 사용자 등록은 tomcat-users.xml 파일에서 한다. 톰캣 설치시 등록한 관리자 계정도 tomcat-users.xml에 등록되어 있다.
- [톰캣 설치 디렉터리\conf\디렉터리]에서 찾을 수 있으며 이클립스의 경우 등록된 톰캣 서버 목록에서 설정 파일을 찾을 수 있다. 여기서는 이클립스에 등록된 톰캣 설정 파일을 사용한다.



[그림 9-4] tomcat-users.xml 파일 위치

03. 아파치 톰캣 시스템 관리

- 사용자 설정을 하기에 앞서 role에 대한 개념을 이해해야 한다.
- role이란 특정 사용자에게 부여할 역할로 사용 자별로 접근할 수 있는 리소스를 통제하려고 여러 권한을 그룹으로 묶어 놓은 것이다.
- 새로운 사용자를 등록하고 권한을 부여하기 위해서는 tomcat-users.xml의 <tomcat-users> </tomcat-users> 사이에 다음과 같이 작성 한다.

```
<role rolename="webadmin" />
<user username="jspbook" password="1234" roles="webadmin" />
```

- 등록된 사용자는 특정 리소스에 대해 사용자 인증을 요구하도록 설정 할 수 있다.
- 사용자 인증은 WEB-INF/web.xml에 추가적인 설정이 필요하다.
- <security-constraint>는 리소스에 대한 인증 설정이고 <login-config>는 로그인 인증을 어떤 방식으로 처리할 것인지에 대한 설정이다.

03. 아파치 톰캣 시스템 관리

- 다음은 ch03 폴더 아래의 모든 요청에 대해 로그인을 요청하는 web.xml 설정이다.

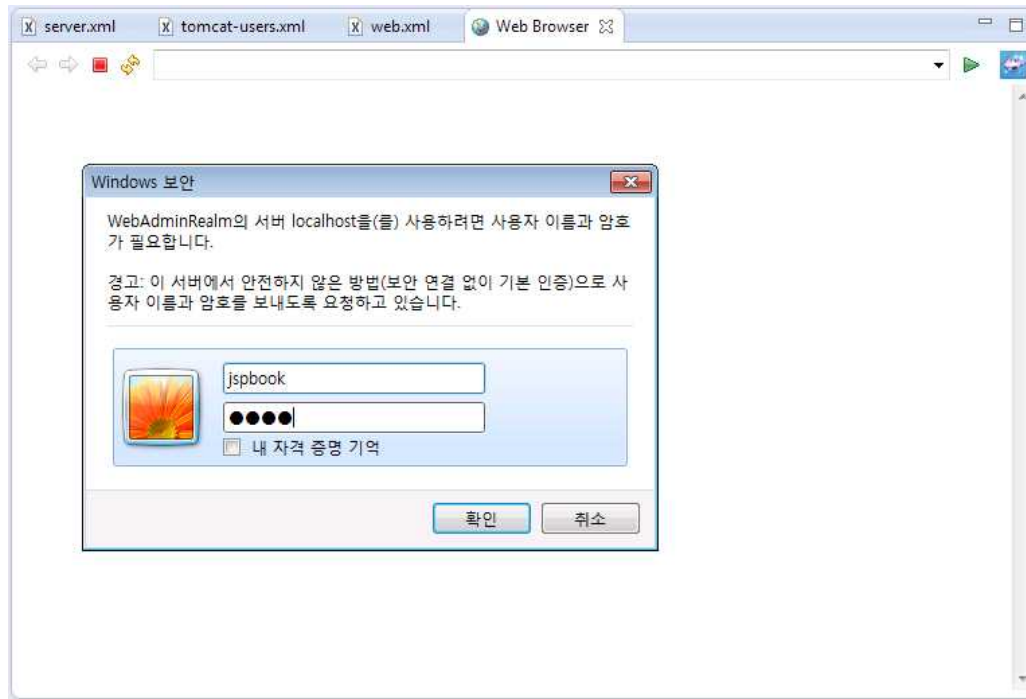
```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>jspbook auth</web-resource-name>
    <url-pattern>/ch03/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>webadmin</role-name>
  </auth-constraint>
</security-constraint>
```

- /ch03/* URL 패턴에 대해 즉, <http://localhost:8080/jspbook/ch03/HelloWorld.jsp> 와 같은 요청에 대해 사용자 인증을 적용하라는 뜻이고 role 은 webadmin 을 적용한다는 의미 이다.
- 다음 설정은 BASIC 사용자 인증 즉, tomcat-usrs.xml 에 등록된 사용자를 웹 브라우저에서 제공하는 로그인 창을 이용해 인증하겠다는 의미가 된다.

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>WebAdminRealm</realm-name>
</login-config>
```

03. 아파치 톰캣 시스템 관리

- 설정을 테스트 하기 위해서는 ch03/HelloWorld.jsp 를 이클립스에서 실행하면 된다.
- 로그인 요청 창에서 등록된 사용자 정보를 입력해야만 실행 결과를 볼 수 있다.

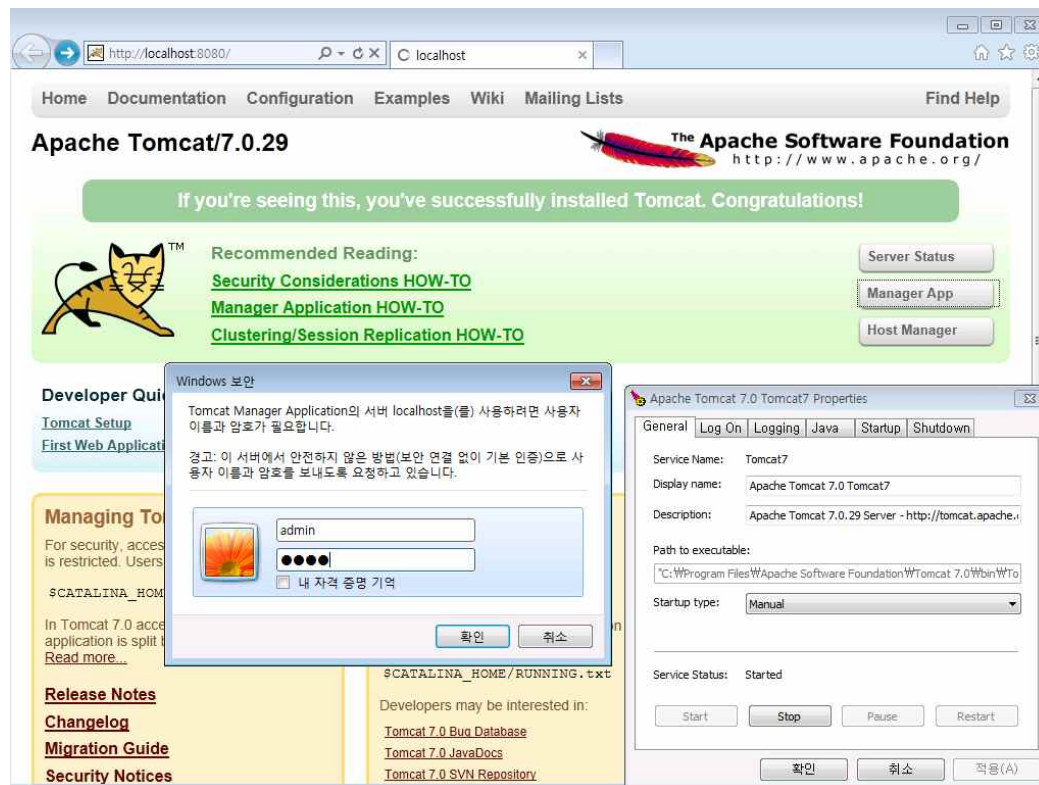


[그림 9-5] tomcat-users.xml 파일 위치

03. 아파치 톰캣 시스템 관리

2. 아파치 톰캣 관리자 모드

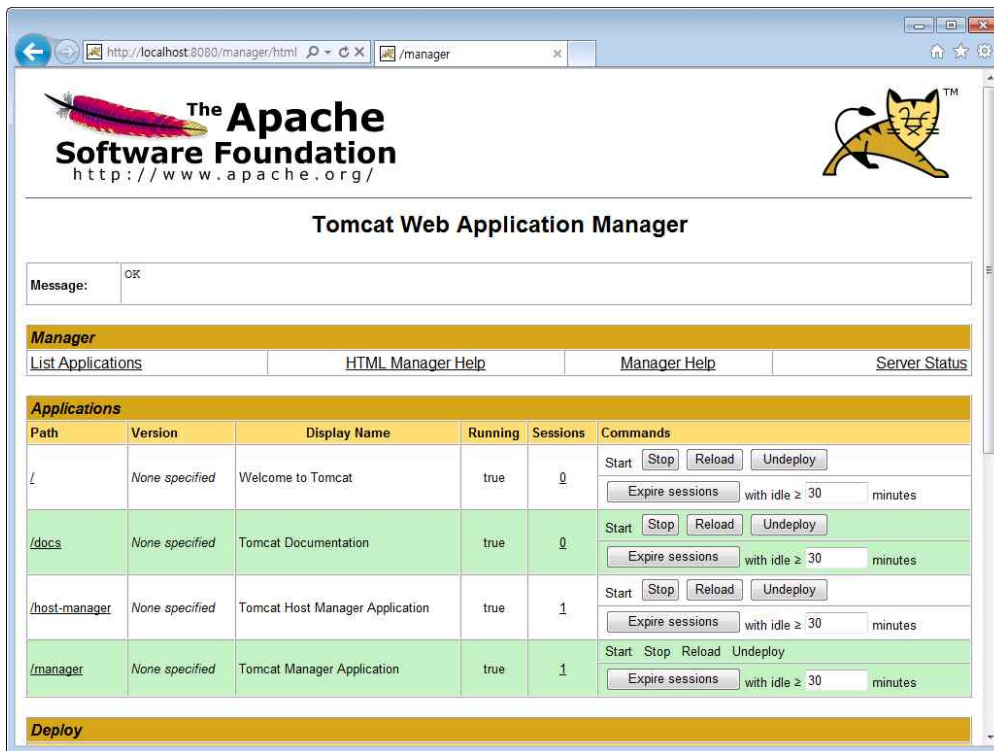
- Tomcat Manager는 톰캣에서 기본적으로 제공하는 웹 기반의 관리도구다.
- 이클립스에 개발용으로 등록된 톰캣은 관리자 모드를 사용할 수 없고 윈도우에 설치된 톰캣을 통해서만 관리자 모드를 사용할 수 있다.
- 이클립스에서 톰캣이 실행중이라면 종료하고 윈도우에서 톰캣을 실행하고 다음 주소로 접속 한다.
- Tomcat Manager 접속 주소 : <http://localhost:8080/manager/html>



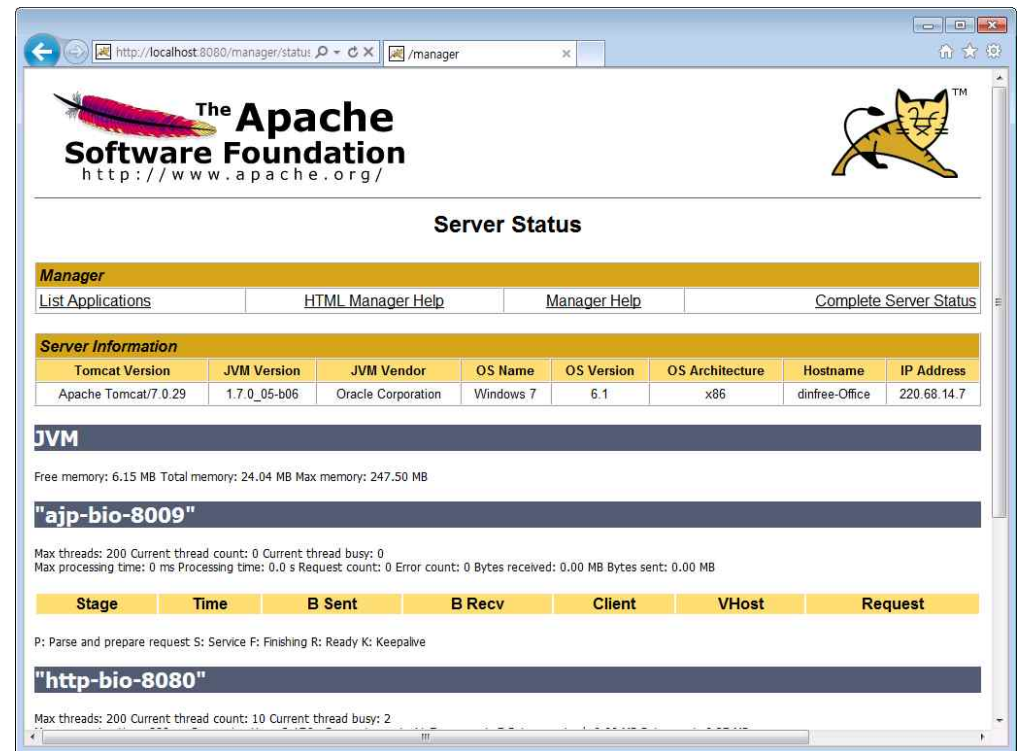
[그림 9-6] Tomcat Manager 로그인 화면

03. 아파치 톰캣 시스템 관리

- Tomcat Manager는 4개의 메뉴로 구성되어 있는데 Help를 제외하면 List Applications 와 Server Status 로 구성되어 있다.
- List Applications는 현재 서버에 설치된 애플리케이션 리스트를 보여주는 부분과 웹 애플리케이션을 신규로 등록하기 위한 Deploy 섹션으로 구분된다. Deploy는 다음 절에서 자세히 살펴본다.
- Server Status에는 현재 톰캣 서버의 각종 정보와 연결된 세션 및 서버 운영 상태를 모니터링 할 수 있다.



[그림 9-7] Tomcat Manager 메인 화면 ①

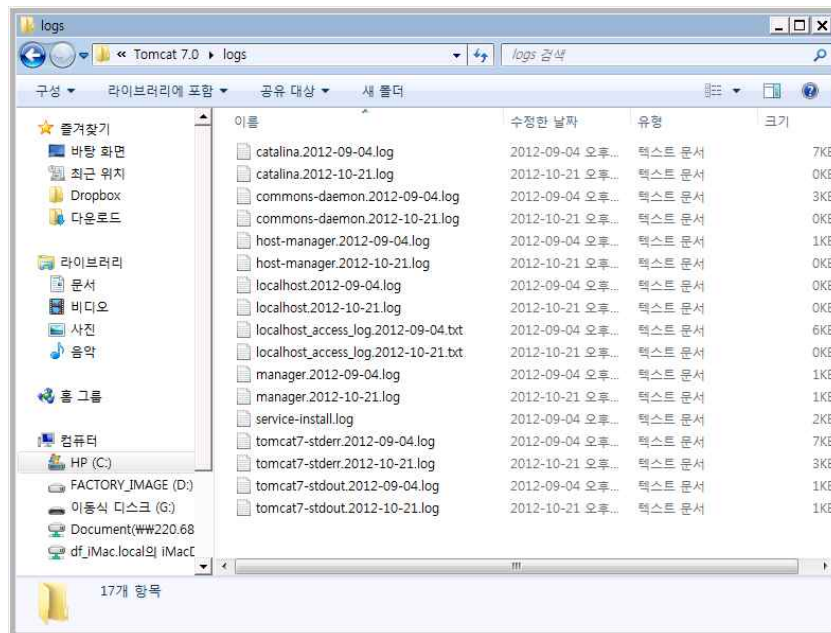


[그림 9-8] Tomcat Manager 메인 화면 ②

03. 아파치 톰캣 시스템 관리

3. 로그 파일 관리

- 시스템 관리에 있어 가장 중요한 부분 중 하나는 로그 파일에 대한 관리 기능이다.
- 로그 파일은 서버의 모든 운영기록, 애플리케이션 오류 등을 파일 형태로 저장하는 것으로 시스템에 문제가 발생했거나 애플리케이션 오류 추적, 웹 서버에 대한 사용자 요청 분석 등 다양한 목적으로 활용된다.
- 여기서는 톰캣 관리 모드와 마찬가지로 이클립스가 아니라 윈도우에 설치된 톰캣을 기준으로 한다. 톰캣 로그 디렉터리는 [톰캣 설치 디렉터리\logs]다.
- 로그파일들의 기본적인 구조는 파일_이름.날짜.log 이다.
- 각각의 로그파일은 텍스트 파일이므로 메모장 등에서 확인 할 수 있다.



[그림 9-9] 로그 파일 목록

03. 아파치 톰캣 시스템 관리

[표 9-2] 주요 로그 파일 목록

파일명	내용
catalina	톰캣의 메인 로그 파일이다. 이클립스 콘솔에서 보던 내용이 담겨 있다. 시스템 운영, 관리 애플리케이션 오류 모니터링에 활용한다.
host-manager	톰캣 관리자 모드 중 하나인 host-manager에서 발생하는 로그가 저장되어 있다.
localhost-access_log	서버에 요청한 리소스 정보가 들어 있다. 사용자 분석, 서버 이용량 분석 등에 활용한다.
manager	톰캣 관리자 모드 사용 정보가 기록되어 있다.
tomcat7-stderr	톰캣의 표준 오류 메시지들을 기록한 파일이다.
tomcat7-stdout	톰캣의 표준 출력 메시지들을 기록한 파일이다.

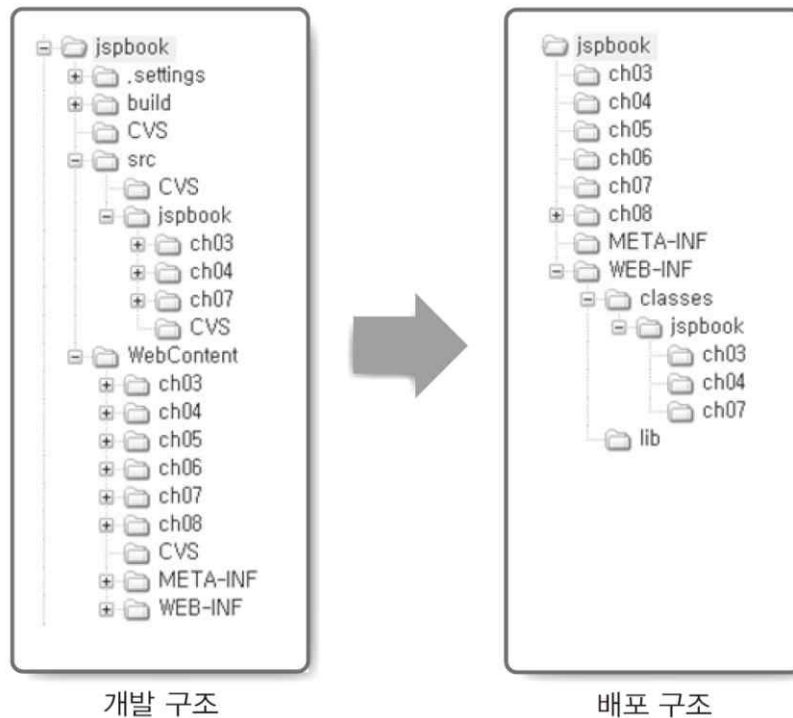
- 웹 서버 접속 현황을 관리하는 localhost-access_log 파일은 날짜별로 생성되며 내용은 다음과 같다.

```
0:0:0:0:0:0:1 - admin [21/Oct/2012:14:16:21 +0900] "GET /manager/images/asf-  
logo.gif HTTP/1.1" 304 -  
0:0:0:0:0:0:1 - admin [21/Oct/2012:14:16:21 +0900] "GET /manager/images/asf-  
logo.gif HTTP/1.1" 304 -  
0:0:0:0:0:0:1 - - [21/Oct/2012:14:39:45 +0900] "GET /docs/ HTTP/1.1" 200 15125
```

04. 웹 애플리케이션 배포하기

1. 웹 애플리케이션 개발

- 웹 애플리케이션은 개발 → 패키징 → 배포 및 실행 → 관리의 과정으로 개발되고 운영된다.
- 개발 환경의 디렉토리 구조와 웹 애플리케이션 디렉토리 구조가 꼭 일치하는 것은 아니며 구조가 달라도 상관 없다. 다만 애플리케이션 배포를 위해서는 이를 웹 애플리케이션 구조로 다시 패키징 하는 작업이 필요하다. Ant 나 Maven 등의 빌드 도구를 사용하면 편리하다.



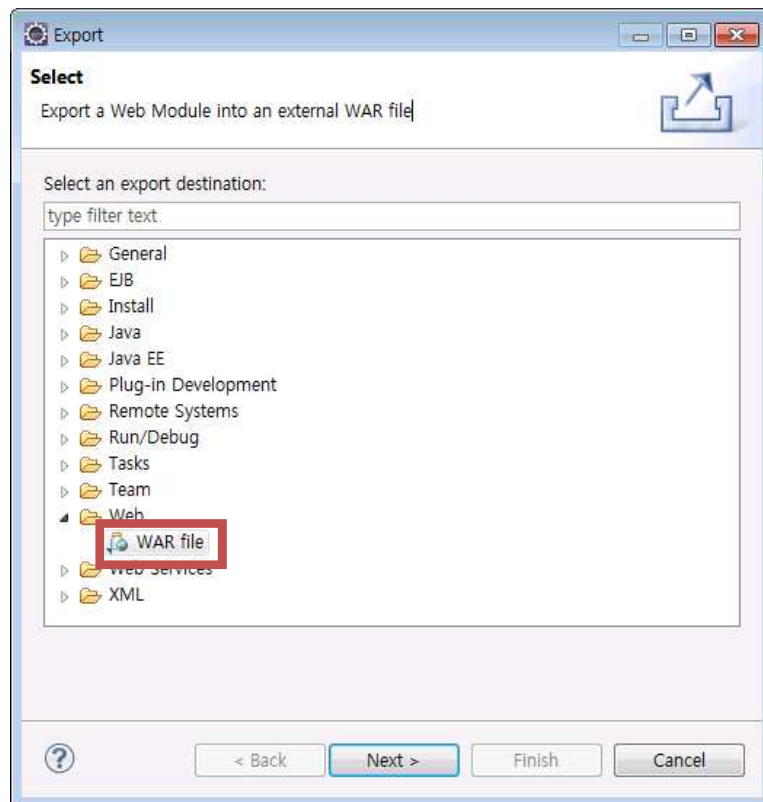
[그림 9-10] 개발 디렉터리 구조와 배포될 디렉터리 구조

04. 웹 애플리케이션 배포하기

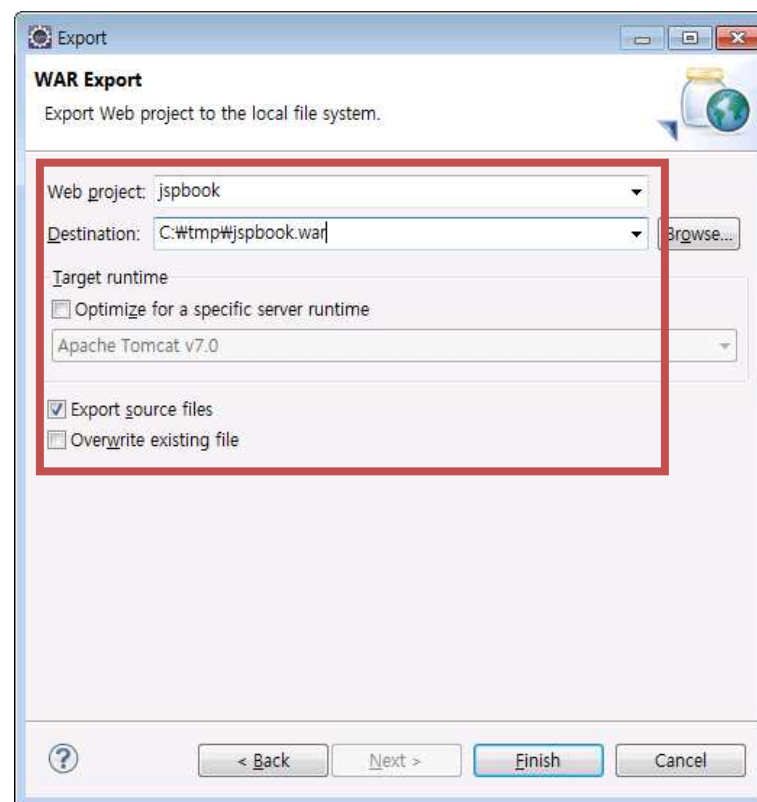
2. 웹 애플리케이션 패키징

- 웹 애플리케이션은 패키징은 이클립스에서 제공하는 Export 기능을 사용하며 패키징 된 파일은 웹 애플리케이션 아카이브 포맷인 .war 파일로 만들어 진다.

❶ 이클립스에서 [File] → [Export]를 선택하면 [그림 9-11]과 같이 유형을 선택하는 화면이 나온다. 여기에 서 'Web' → 'War file'을 선택하도록 한다. 그리고 <Next> 버튼을 눌러 옵션을 지정한다.



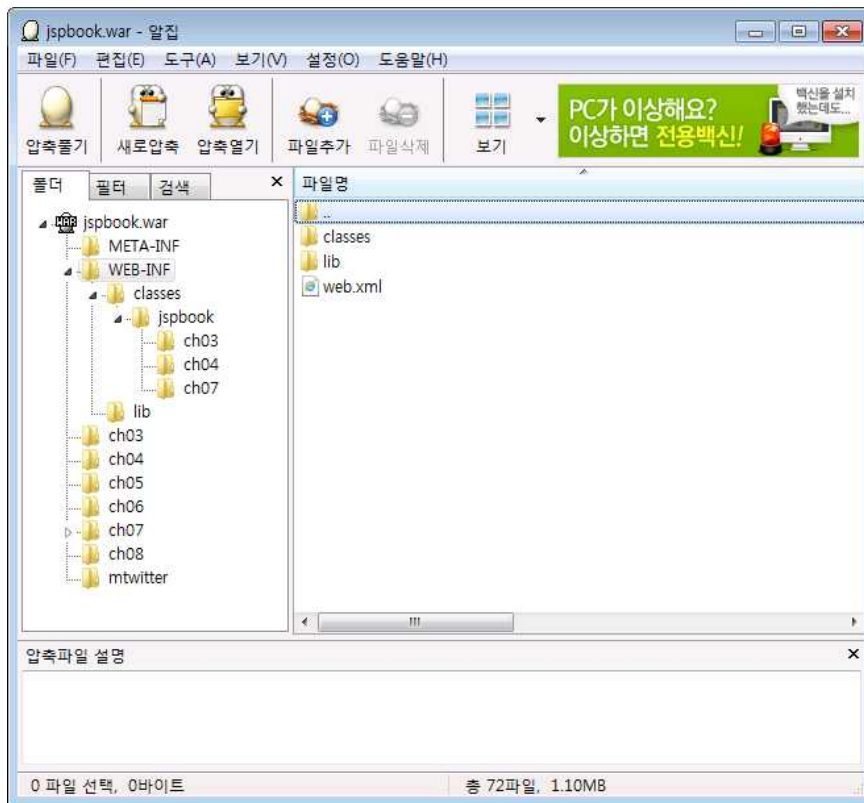
[그림 9-11] 내보내기 유형 선택



[그림 9-12] 설정 선택

04. 웹 애플리케이션 배포하기

② <Finish> 버튼을 누르면 [c:\WTmp] 폴더에 jspbook.war 파일이 생성된다.



[그림 9-13] WAR 파일 내용

04. 웹 애플리케이션 배포하기

3. 배포와 실행

- 웹 애플리케이션의 배포, 즉 설치하는 방법에는 컨테이너가 종료된 상태에서 디렉터리 구조에 WAR 파일을 직접 복사하거나 압축을 해제하는 방법, 실행 중인 서버에 동적으로 설치하는 방법이 있다.
- 톰캣을 종료하면 실행 중인 모든 서비스가 종료되므로, 대부분의 경우 서버를 종료하기보다는 실행 중인 서버에 설치하는 것이 유리하다.
- 실행 중인 서버에 설치하는 경우 해당 컨테이너에서 제공하는 배포 툴이 있어야 한다. 톰캣의 경우 앞서 살펴본 Tomcat Manager라는 툴을 제공하고 있다.

■ 디렉터리 구조에 직접 설치

❶ 디렉터리 구조에 직접 설치하는 경우 톰캣의 설치 디렉토리를 이용한다. 다음 디렉토리에 export 된 jspbook.war 파일을 복사 하도록 한다.

- c:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\jspbook.war

❷ 윈도우에서 톰캣 실행

❸ 톰캣이 실행되면 자동으로 war 파일의 압축이 해제되고 디렉터리에 파일들이 생성됨.

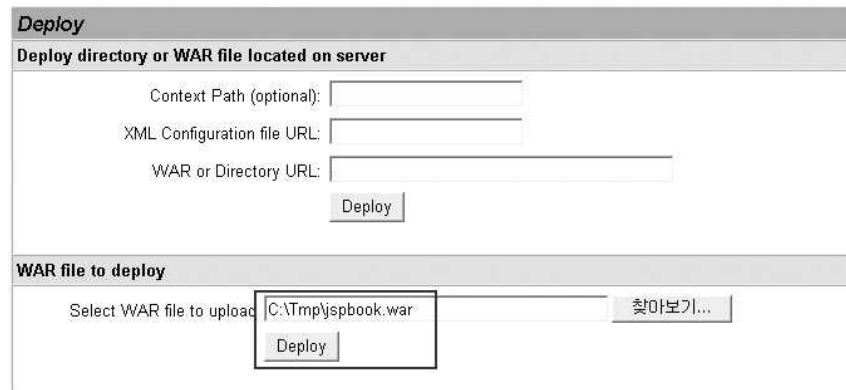
❹ 다음과 같이 브라우저에 URL 을 입력해 접속해 정상 동작 확인.

- <http://localhost:8080/jspbook/ch03/HelloWorld.jsp>

04. 웹 애플리케이션 배포하기

■ Tomcat Manager를 이용한 설치

- 앞에서 디렉토리에 설치 했다면 톰캣을 종료 하고 jspbook.war 와 jspbook 폴더를 삭제 한 후 톰캣을 다시 시작 한다.
- <http://localhost:8080/manager> 로 접속해 로그인 하고 메인 화면 중앙의 Deploy 섹션에서 "찾아보기" 버튼을 클릭해 export 한 jspbook.war 파일을 선택후 Deploy 버튼을 클릭함.



The screenshot shows the 'Deploy' section of the Tomcat Manager interface. It has two main parts: 'Deploy directory or WAR file located on server' and 'WAR file to deploy'. The first part has input fields for 'Context Path (optional)', 'XML Configuration file URL', and 'WAR or Directory URL', with a 'Deploy' button below. The second part has a 'Select WAR file to upload' text, a file input field containing 'C:\Tmp\jspbook.war', a '찾아보기...' (Browse...) button, and a 'Deploy' button.

[그림 9-14] WAR 파일 선택

- 정상적으로 deploy가 되었으면 다음과 같이 jspbook이 등록된 것을 확인할 수 있다.

Applications				
Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/jspbook	jspbook	true	0	Start Stop Reload Undeploy
/manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy

[그림 9-15] jspbook 애플리케이션 설치 확인

04. 웹 애플리케이션 배포하기

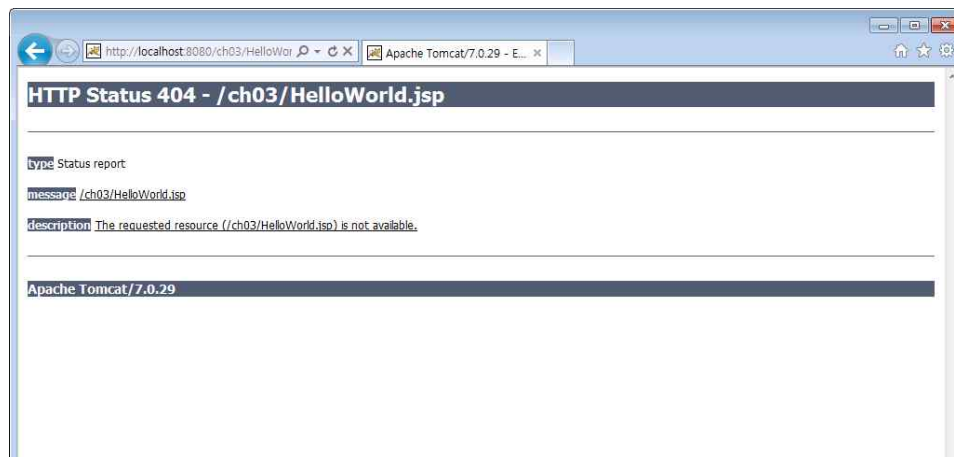
4. 관리

- 설치된 웹 애플리케이션은 상황에 따라 잠시 운영을 중단 하거나 재시작 하는 경우가 발생 한다.
- 이 때 Tomcat Manager에서 해당 작업을 수행할 수 있다.
- Application List 의 애플리케이션 목록 테이블 오른쪽에 있는 Start, Stop, Reload, Undeploy 메뉴를 사용하면 된다.

Applications				
Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start <u>Stop</u> Reload Undeploy
/host-manager	Tomcat Manager Application	true	0	Start <u>Stop</u> Reload Undeploy
/jspbook	jspbook	true	1	Start <u>Stop</u> Reload Undeploy
/manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/tomcat-docs	Tomcat Documentation	true	0	Start <u>Stop</u> Reload Undeploy

[그림 9-16] 애플리케이션 종료

- 애플리케이션을 종료 한 다음 HelloWorld.jsp를 다시 실행해 보면 다음과 같이 찾을 수 없다는 메시지를 볼 수 있다.



[그림 9-17] 애플리케이션 삭제 확인



프로젝트로 배우는 자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음