JEUS Web Server 안내서

Document info
Document name: JEUS Web Server 안내서
Document date: 2005-06-06
Manual release version: 3
Software Version: JEUS 5

차 례

그림 목차

표 목차

매뉴얼에 대해서

매뉴얼의 대상

본 매뉴얼은 TmaxSoft 의 JEUS 제품군에 포함된 JEUS Web Server 를 설정, 관리하는 사람들에게 필요한 정보를 담고 있다.

JEUS Web Server 는 JEUS Web Application Server 와의 연동을 위해서, 특별히 고안되어 개발된 Web Server 이다.

매뉴얼의 전제 조건

본 매뉴얼을 이해하기 위해서는 Web Server 와 World Wide Web, TCP/IP, HTTP 에 대한 기초적인 이해만 있으면 되고, 특별한 전문지식이 요구되지는 않는다. 그렇지만 JEUS Web Server 를 UNIX 환경에서 운영하고자 하는 경우라면, UNIX 사용에 관련한 기본적인 지식이 요구된다.

만약, JEUS Web Container 와의 연동을 위해서 JEUS Web Server 를 사용하려고 한다면, 이 매뉴얼을 읽는 전/후에, 간단하게라도 JEUS Web Container 안내서를 읽어봐야 할 것이다.

매뉴얼의 구성

본 매뉴얼은 열 개의 장(Chapters)과 다섯 개의 부록으로 구성되어져 있다. 중요한 열 개의 장(Chapters)은 다음과 같다.

1. 소개
2. 따라 하기
3. JEUS Web Server 의 개요
4. JEUS Web Server 설정
5. JEUS Web Server 튜닝

6. JEUS Web Server 제어
7. JEUS Web Server 관리
8. JEUS Web Server 보안 설정
9. JEUS 와의 연동
10. 결론

다섯 개의 부록은 다음과 같다.

A wsadmin: JEUS Web Server 관리툴 레퍼런스

B wscfl: JEUS Web Server 컴파일 툴 레퍼런스

C wsmkpw: JEUS Web Server 인증 툴 레퍼런스

D JEUS Web Server 환경설정 예

E JEUS Web Server 환경설정 레퍼런스

F WSMain.xml XML 구성 레퍼런스

JEUS 및 JEUS Web Server 를 처음 접하는 관리자인 경우, JEUS 와의 연동 및 JEUS Web Server 의 적절한 설정을 위해서, 적어도 한 번은 주요 장 (1 장 ~ 10 장)을 읽어야 한다. 이 주요 장들은 순서대로 읽어야 한다.

부록에서 제공하는 내용은, 본문에서 간단하게 설명했던 내용들에 대해서, 기술적으로 보다 상세하게 다시 설명한다. 그러므로 이 부록들은 JEUS 및 JEUS Web Server 에 대해서 경험이 많은 관리자들이 좋은 참고 자료로 사용할 수 있다.

관련 매뉴얼

본 매뉴얼과 관련 있는 것들은 다음과 같다.

- JEUS Web Container 안내서.
- JEUS Server 안내서.

일러두기

표기 예	내용
텍스트	본문, 12 포인트, 바탕체 Times New Roman
텍스트	본문 강조
CTRL+C	CTRL 과 동시에 C 를 누름
`public class myClass { }`	Java 코드
`<system-config>`	XML 문서
참조: / 주의:	참조 사항과 주의할 사항
Configuration 메뉴를 연다	GUI 의 버튼 같은 컴포넌트
JEUS_HOME	JEUS 가 실제로 설치된 디렉토리 예)c:\jeus
`jeusadmin nodename`	콘솔 명령어와 문법
[파라미터]	옵션 파라미터
< xyz >	'<'와 '>' 사이의 내용이 실제 값으로 변경됨
|	선택 사항. 예) A|B: A 나 B 중 하나
…	파라미터 등이 반복되어서 나옴
?, +, *	보통 XML 문서에 각각 "없거나, 한 번", "한 번 이상", "없거나, 여러

	번”을 나타낸다.
. . .	XML 이나 코드 등의 생략
<<*FileName.ext*>>	코드의 파일명
그림 1.	그림 이름이나 표 이름

연락처

Korea
Tmax Soft Co., Ltd
18F Glass Tower, 946-1, Daechi-Dong, Kangnam-Gu, Seoul 135-708
South Korea
Tel: 82-2-6288-2114
Fax: 82-2-6288-2115
Email: info@tmax.co.kr
Web (Korean): http://www.tmax.co.kr

USA
Tmax Soft, Inc.
560 Sylvan Ave, Englewood Cliffs NJ 07632
USA
Tel: 1-201-567-8266
FAX: 1-201-567-7339
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com

Japan
Tmax Soft Japan Co., Ltd.
6-7 Sanbancho, Chiyoda-ku, Tokyo 102-0075
Japan
Tel: 81-3-5210-9270
FAX: 81-3-5210-9277
Email: info@tmaxsoft.co.jp
Web (Japanese): http://www.tmaxsoft.co.jp

China
Beijing Silver Tower, RM 1507, 2# North Rd Dong San Huan,
Chaoyang District, Beijing, China, 100027
Tel: 86-10-6410-6148
Fax: 86-10-6410-6144
E-mail : info@tmaxchina.com.cn
Web (Chinese): http://www.tmaxchina.com.cn

1 소개

JEUS Web Server 는 차원이 다른 성능과 안정성을 갖춘 차세대 웹 서버로서, World Wide Web 환경에서, HTTP 프로토콜을 통해서 전달된 Browser 의 Request 를 처리하는 웹 서버이다. JEUS Web Server 는 세계 유명 웹 서버 보다 월등히 우수한 성능을 보장할 뿐만 아니라, 기존 웹 서버가 가지고 있는 한계점들을 완벽하게 개선함으로써, Web 환경을 구축하는데 있어 고성능과 안정성이라는 두 가지 효과를 모두 얻을 수 있다.

2 따라 하기

다음에 나오는 간단한 절차를 통해서, JEUS Web Server 를 처음 접하는 사용자들이 바로 설치하여 기동할 수 있도록 안내할 것이다.

아래의 예에서 우리는 노드명을 "webmain"으로 사용할 것이다. 필요하다면, 이 노드명을 자신의 컴퓨터이름(HOSTNAME)으로 변경하여 사용하도록 한다.

1. 첫째, JEUS Web Server 가 시스템에 제대로 설치되었다면, PATH 를 비롯한 시스템 변수들도 설정되어 있을 것이다. JEUS Web Server 의 설치에 대해서 보다 자세한 설명을 원한다면, JEUS 설치 안내서를 참조하기 바란다.

2. JEUS Web Server 의 환경 파일이 설정되어 있는지 확인한다. JEUS_WSDIR\config\ 디렉토리 아래의 ws_engine.m 파일이 다음과 같이 되어 있어야 한다.

<<*ws_engine.m*>>

```
*DOMAIN
JEUS_Web_Server

*NODE
webmain   WEBTOBDIR="C:/jeus/webserver",
          SHMKEY = 54000,
          DOCROOT="C:/jeus/webserver/docs",
          PORT = "8080",
          LOGGING = "log1",
          ERRORLOG = "log2",
          JSVPORT = 9900,
          HTH = 1

*SVRGROUP
htmlg   NODENAME = webmain, SVRTYTE = HTML
cgig    NODENAME = webmain, SVRTYPE = CGI
ssig    NODENAME = webmain, SVRTYPE = SSI
jsvg    NODENAME = webmain, SVRTYPE = JSV
```

```
*SERVER
html    SVGNAME = htmlg, MinProc = 1, MaxProc = 2
cgi     SVGNAME = cgig, MinProc = 1, MaxProc = 2
ssi     SVGNAME = ssig, MinProc = 1, MaxProc =2
MyGroup SVGNAME = jsvg, MinProc = 25, MaxProc = 30

*URI
uri1    Uri = "/cgi-bin/", Svrtype = CGI
uri2    Uri = "/examples/",Svrtype = JSV

*ALIAS
alias1   URI = "/cgi-bin/",
         REalPath = "C:/jeus/webserver/cgi-bin/"

*LOGGING
log1    Format = "DEFAULT",
        FileName = "C:/jeus/webserver/log/access.log",
        Option = "sync"
log2    Format = "ERROR",
        FileName = "C:/jeus/webserver/log/error.log",
        Option = "sync"

*EXT
htm    MimeType = "text/html", SvrType = HTML
```

3. JEUS Web Server 의 환경파일 컴파일 툴(wscfl)을 이용해서, ws_engine.m 파일을 컴파일한다.

   ```
   wscfl –i ws_engine.m
   ```

   보다 자세한 설명은 4 장 “JEUS Web Server 설정”을 참조하기 바란다.

4. “JEUS_WSDIR\config\” 디렉토리에 바이너리 파일인 wsconfig 가 생성되었는지 확인한다..

5. 지금까지의 설정들을 확인했으면, JEUS Web Server 를 기동하기 위해 , 명령 프롬프트에서 “wsboot”라고 입력한다.

6. Browser 를 열고, 다음과 같이 URL 을 입력한다.

`http://[머신의 IP]:[80 번트로 하지 않았다면 포트 번호]/`

[그림 1]과 같이, JEUS Web Server Test Page 가 보여질 것이다.

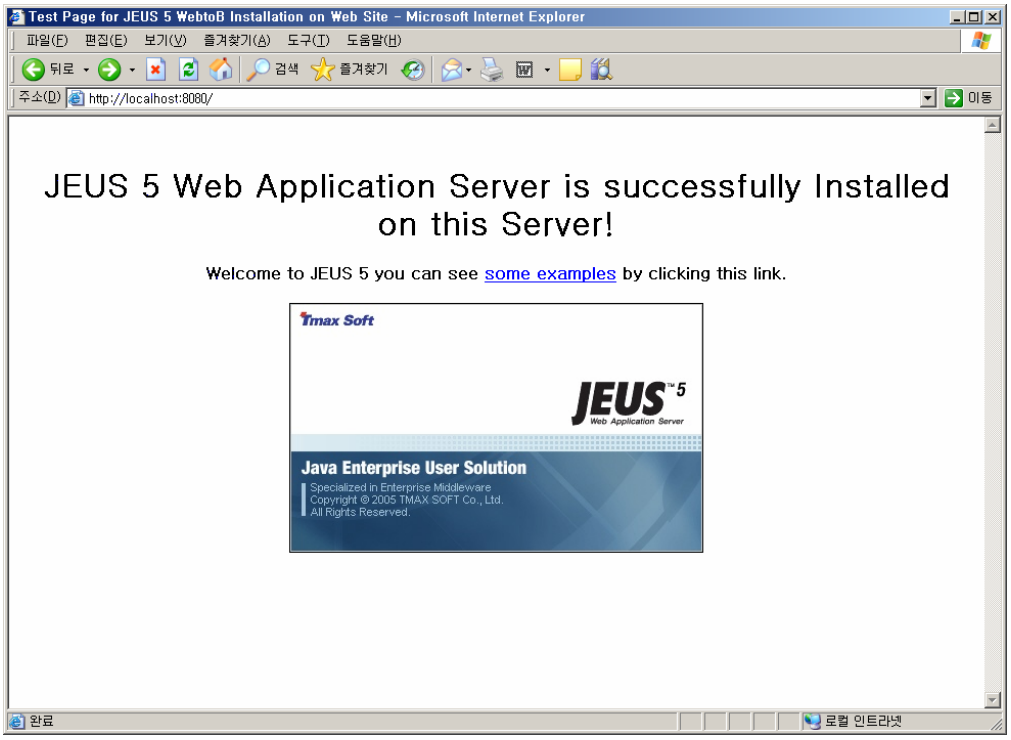

그림 1. JEUS Web Server Test Page

7. JEUS Web Server Admin Tool 을 띄우기 위해, 새로운 명령 프롬프트 창을 열어서 “wsadmin”을 입력한다.

8. 해당 창에서 “help”나 “h”를 입력하면, JEUS Web Server 를 모니터링 하거나 동작을 제어할 수 있는 명령어의 리스트를 확인할 수 있다.

9. 관리자 명령어들을 실제 실행하여 확인하였으면, 다음으로, “quit”를 입력하여, 해당 JEUS Web Server Admin Tool 을 빠져 나온다.

10. “wsboot”를 실행한 명령 프롬프트 창으로 바꿔서, “wsdown”을 입력하면, JEUS Web Server 가 내려가게 된다.

11. 이제 JEUS Web Server system 전체가 종료되었다.

이것으로 “따라 하기”를 통한 설명은 마치고, 계속해서 보다 깊고 상세하게, 여러 가지 툴과 환경설정에 관련한 설명이 이어질 것이다.

만약, 위에서 진행한 각 단계 별로 테스트가 되지 않는다면, JEUS 설치 안내서와 JEUS Server 안내서를 참조해서 현재의 설정을 확인해 보기 바란다.

3 JEUS Web Server 의 개요

3.1 소개

이번 장에서는 JEUS Web Server 의 기본적인 구조와 개념, 그리고 특징들에 대해서 설명한다.

3.2 구조적인 개념

JEUS Web Server 는 기존 웹 서버와는 다른 구조를 가지고 있다. Apache 로 대변되는 기존 웹 서버들은 대부분 NCSA 사의 Httpd 의 구조를 가지고 있다. 이는, 사용자가 그리 많지 않던 환경 하에서 이용되던 것으로, 사용자의 증가에 유연하게 대처하지 못하는 단점을 지니고 있다.

JEUS Web Server 에선 이런 문제를 가장 먼저 해결하고자 하였다. 따라서, 최우선적으로 고려한 것이 사용자가 계속 증가하는 경우에 대한 대처 방식이었다.

기존의 웹 서버들은 사용자의 Request 가 들어오면, 일대일 방식으로 연결을 맺는 구조로 되어있다. 따라서 만약 사용자가 증가하는 경우, 결국 Server Process 가 그 수만큼 동시에 늘어나야 Service 를 할 수 있었다. 그러나, 이러한 방식은 사용자가 증가하면 할수록, Server 에 걸리는 부하도 따라서 증가하기 때문에 많은 문제점을 야기한다.

물론, 사용자가 늘어나면 Server 에 걸리는 부하가 적을 수는 없다. 더 많은 Memory 와 CPU Overhead 를 감수하여야 하는 것은 당연한 일이다. 하지만, 사용자가 늘어난다고 해서 Server 에 걸리는 부하가 이와 비례해서 계속 크게 늘어나, 더 많은 Hardware 를 추가로 구입하여야 하는 일이 발생한다면, 이는 적절한 해답이 되지 못한다. 즉, 사용자가 늘어난다고 해도 Server 자체에서 적절히 대응하여 일정 수준까지는 무리 없이 Service 를 가능하게 하는 구조가 되어야 무한하게 확장하는 Web 에 대응할 수 있는 것이다.

그럼 여기서 기존의 웹 서버와 JEUS Web Server 와의 구조에 대해 알아보고 JEUS Web Server 의 구조가 무한히 확장하는 Web 환경에 대응할 수 있는 이유를 알아보자.

초창기 웹 서버인 Httpd 의 구조를 그대로 계승하는 덕에 Apache 는 가장 일반적인 Process per Request 형태를 취하고 있다. 즉, 사용자 Request 마다 Process 가 발생하여 처리하는 구조인 것이다. 따라서, 앞에서 언급한 대로 사용자가 증가하면, Process 도 같이 비례하여 증가하게 되므로, 사용자 수가 일정 수 이상이 되면, Service 에 많은 무리가 생긴다. 이는 각 시스템 마다 Process 의 수가 한정되어 있고, Process 가 늘어나면 너무 많은 Memory 를 차지하는 등의 Overhead 도 같이 늘어나기 때문이다.

이로 인하여 새로운 상용 웹 서버들이 등장하면서, 급속도로 확장하고 있는 Web 환경에 대처하기 위하여 새로운 구조를 채택하였다. 가장 일반적인 것이 Multi-Thread 기술을 이용한 것이다. 이는 Process 보다 Overhead 가 적은 Thread 를 이용하여 Service 를 하는 것으로 기존 Process 방식에 비하여 월등히 좋은 성능을 나타내었다. Thread 는 Process 내부에서 여러 개를 동시에 이용할 수 있기 때문에 Overhead 도 적고, 차지하는 Memory 도 Process 방식에 비하여 상당히 적기 때문이다.

그러나, 이러한 방식도 문제를 가지고 있다. Thread 기술이 상당히 우수한 기술임에는 틀림없지만, 내부에서 자신들 간의 영역 다툼으로 인하여 DeadLock 등의 문제가 빈번하게 발생하기 때문이다. 이것은 특히 안정성에 있어 치명적이다. Thread 구조에서 DeadLock 이 발생하면, 자체 Thread 가 Service 를 수행하지 못함은 물론, 자칫하면 전체 시스템이 Down 되어 Service 불능 상태가 될 수 있다. 이는, 성능 뿐 아니라, 안정성도 중요한 요소가 되는 웹 서버에 치명적인 결함이 될 수 있다.

위에서 언급한 두 가지 방식은 서로 장단점이 있다. 성능과 안정성 면에서 서로 양극단을 달리고 있다고 보면 될 정도로 차이가 있다. 하지만, 성능과 안정성 모두 웹 서버에게 있어서는 놓쳐서는 안될 아주 중요한 고려 사항이다. 둘 중의 하나만 충족되지 않아도 원만한 Service 가 되지 않기 때문이다.

TmaxSoft 에서는 JEUS Web Server 를 개발하면서, 성능과 안정성을 모두 충족시키기 위하여 많은 구조를 검토하고 연구하였다. JEUS Web Server 에서는 안정성을 위하여 Thread 구조를 배제하고 Process 구조를 채택하였다. 그리고, 성능을 위하여 각 Process 를 독립적으로 만들지 않고, 각각의 특별한 기능을 하는 Process 를 두어 역할을 분담하였다. 또한, 각 Process 마다 만약의 경우를 대비하여 여분의 보조 Process 를 두는 것을 가능하게 하여, 안정성에 치중하였다.

JEUS Web Server 에서 Service 를 하는 방식은 아래의 그림과 같다.

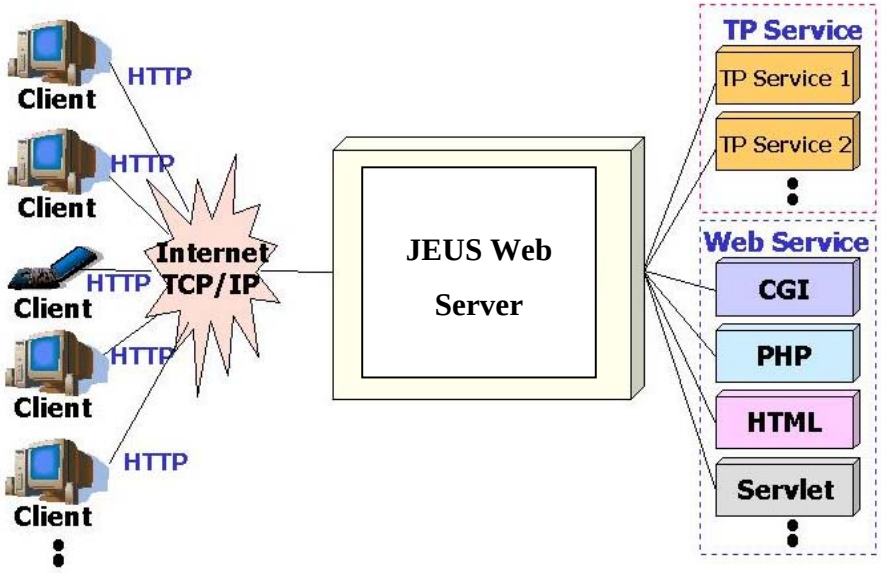

그림 2. JEUS Web Server 의 기본 구조

3.3 특징

이번에 TmaxSoft 에서 새롭게 업그레이드된 JEUS Web Server 의 대표적인 특징을 살펴보면 다음과 같다. 첫번째로 Memory Cashing, Web API, Dynamic Configuration 등 기존의 구 버전에서 갖고 있던 기능들의 향상이 있었으며, Web Application Server 로써의 기능이 대폭 향상되었다는 점이다. TDBC, TSP, Session Management, 타제품과의 통합 등 Web Application Server 가 수행하는 역할을 JEUS Web Server 하나만으로도 충분하게끔 그 기능이 대폭 증강되었다는 점을 들 수 있다.

3.4 JEUS Web Server 의 Tool

다음에 설명되는 툴들은 JEUS Web Server 의 엔진 프로세스 및 서버 프로세스들을 관리하기 위해서 사용된다.

- **wsadmin**: JEUS Web Server system 의 전체적인 관리를 위해서 사용되는 툴로서, 시스템 정보 및 관리자로서의 작업들을 수행할 수 있다.
- **wscfl**: 텍스트 기반의 환경파일을 컴파일해서 바이너리 파일로 변환한다.

- **wsmkpw**: 사용자 인증에 관련된 정보들을 암호화해서 특정 파일을 생성한다.

3.5 JEUS Web Server 의 디렉토리 구조

기본적으로 JEUS Web Server 는 JEUS Application Server 를 설치하게 되면 같이 설치된다. JEUS Web Server 가 설치되는 기본 디렉토리는 JEUS_HOME/webserver 이다. 여기서 "JEUS_HOME"은 JEUS Application Server 의 홈 디렉토리이다.

아래의 [그림 3]이 JEUS Web Server 가 설치된 디렉토리의 구조이다. 그림에서 "JEUS_WSDIR"은 "JEUS_HOME/webserver"로 보면 된다.

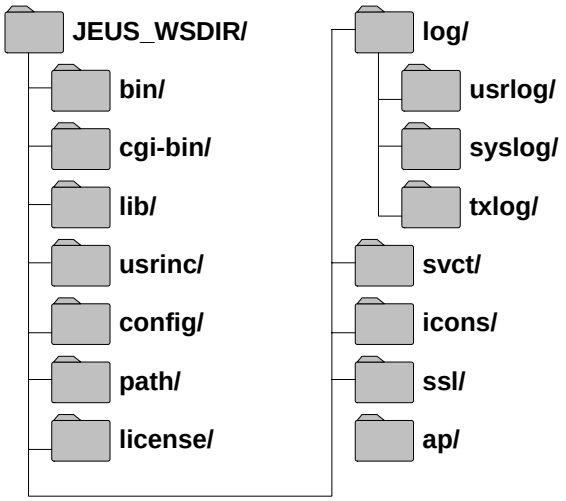

그림 3. JEUS Web Server 디렉토리 구조

JEUS Web Server [그림 3]의 중요 디렉토리들 :

- **bin/:** 실행 파일들이 위치.(wsm, wscfl, wsracd, wsgst, wsboot, wsdown, etc.).
- **cgi-bin/:** CGI 파일이 위치
- **lib/:** Library 파일이 위치
- **usrinc/:** API 의 헤더 파일들이 위치
- **config/:** JEUS Web Server 의 환경파일이 위치
- **path/:** 프로세스간의 내부 통신을 위한 Named-pipe 가 생성

- **docs/:** 기본적으로 설치되는 html 파일이 위치
- **log/:** Log 파일들이 위치
- **svct/:** Web API 의 서비스 테이블이 위치
- **icons/:** DIRINDEX 에서 사용할 아이콘이 위치
- **ssl/:** SSL 관련 파일이 위치
- **ap/:** Application 파일이 위치

3.6 환경 변수

JEUS Web Server 에서 사용하는 환경변수로 ”JEUS_WSDIR”이 있다. 이 환경변수의 값으로는 JEUS Web Server 의 설치 디렉토리가 세팅 되어야 한다. 기본적으로 JEUS 가 설치되는 동안 JEUS_HOME/webserver 로 설정된다. 보다 상세한 정보를 원한다면 “JEUS 설치 안내서”를 참조하기 바란다.

참고 :환경변수 “JEUS_WSDIR”은 JEUS Web Server 의 환경설정 파일에 정의된 “WEBTOBDIR”과 같아야 한다.

4 JEUS Web Server 설정

4.1 소개

본 장에서는 JEUS Web Server 를 구동하는데 필요한 정보를 가지고 있는 설정 파일에 대해서 설명한다. 이는, 본 매뉴얼에서 핵심이 되는 부분이며, 차후의 설명에도 반드시 배경 지식으로 필요한 것이다. 따라서, JEUS Web Server 의 환경파일에 대한 이해가 부족하다면 반드시 이 장을 숙지하여야 한다.

참고 : JEUS Web Server 의 환경설정에 대해서 보다 자세한 정보를 알고 싶다면, 본 매뉴얼의 끝에 있는 부록 F 를 참조하기 바란다.

4.2 환경파일의 개념

4.2.1 JEUS Web Server 의 환경파일이란 무엇인가?

환경파일이란, JEUS Web Server 의 환경 설정에 사용되는 파일로서 실제 JEUS Web Server 기동에 필요한 모든 정보를 제공한다. 즉, JEUS Web Server 를 기동하는데 있어 사용자가 원하는 바를 이 환경파일을 통하여 JEUS Web Server 에게 전달하는 것이다.

JEUS Web Server 는 실제 구동 시에, 이 환경파일에 기입된 정보를 바탕으로 모든 기능을 수행하게 된다. 따라서 사용자가 실제 환경에 맞는 적당한 정보를 기입하면, JEUS Web Server 는 원활하게 기능을 수행할 수 있고, 만약 잘못된 정보를 기입한다면, 사용자가 원하는 대로 작동하지 않거나, 문제가 생기게 된다.

4.2.2 JEUS Web Server 환경파일의 일반적인 형태

JEUS Web Server 의 환경설정 파일은 기본적으로 4 개의 절로 구성된다. DOMAIN, NODE, SVRGROUP, 그리고 SERVER 가 그것이다. 각 절의 기본적인 형태는 다음과 같다.

```
*절
 이름          항목 1= ..., 항목 2= ..., ...
```

아래는 JEUS Web Server 환경파일의 간단한 예이다.

<<*http.m*>>

```
*DOMAIN
JEUS_Web_Server1
*NODE
webmain     WEBTOBDIR=”/usr/local/webserver/”, HTH = 4,
            SHMKEY = 77990, PORT = “9500”,
            DOCROOT=”/usr/local/webserver/docs”
*SVRGROUP
svg1        NODENAME = webmain, SVRTYPE = HTML
svg2        NODENAME = webmain,  SVRTYPE = CGI
*SERVER
html        SVGNAME = svg1, MinProc = 1, MaxProc = 2
cgi         SVGNAME = svg2, MinProc = 1, MaxProc = 2
```

각각의 절의 이름은 별표(*)로 시작된다. 즉, *로 시작되는 항목은 JEUS Web Server 가 절로 인식한다. 절의 이름으로는 DOMAIN, NODE, SVRGROUP, SERVER 가 사용되며, 이들은 모두 대문자이다. 절의 정의 순서는 고정되어 있지 않다. 따라서 어느 절이 먼저 정의되어도 상관없다. 그러나, Tree 형태로 구성되어 있기 때문에 가급적 순서를 지켜 주는 것이, 나중의 편의성을 위하여 좋다.

또한 절의 정의는 내용별로 나누어 한 번 이상 나타날 수 있다. 즉, 하나의 절이 여러 번으로 나누어 정의될 수 있다. 예를 들어, Multi Node 에서 Node 별로 해당 Server Group, 업무처리 Server Process 등을 분리하여 정의하고자 할 경우, 아래와 같은 JEUS Web Server 의 환경파일 구성이 가능하다. 이 경우, 동일한 이름으로 같은 내용이 반복 정의 되어서는 안 된다.

<<*http.m*>>

```
*DOMAIN
 JEUS_Web_Server1
*NODE
 node1            .......
*SVRGROUP
 ........
*SERVER
 ........
*NODE
```

```
 node2              .......
*SVRGROUP
 ........
```

각 절의 이름이나 절의 하위 개체 명은 반드시 첫 번째 칸에서 시작되어야 한다. 그 이후 항목들은 윗 줄에 이어지는 내용이라 하더라도 반드시 공백을 두고 시작되어야 한다.

각 절의 하위 개체 정의시, 하나의 개체(즉 Node, Server Group, Server)를 정의하는 항목들은 반드시 콤마(,)로 연결되어야 한다. 항목 정의가 콤마로 끝났을 경우, 다음 항목도 현재 대상 개체에 대한 내용으로 적용되지만, 콤마로 이어지지 않는 경우 현재 대상 개체의 정의는 끝난 것으로 인식된다.

각 절의 항목들은 '항목이름 = 값'의 형태로 정의된다.

각 항목은 다음의 4 가지 형태 중 하나의 값으로 정의될 수 있으며, 항목 이름은 값으로 사용될 수 없다.

항목의 데이터 타입:

- **Numeric**: 숫자
- **String**: abc 형태의 문자열.
- **Literal**: "abc" 형태의 문자열 (""에 주의).
- **Y / N**: Yes, No 형태

각 절의 항목들을 다음과 같은 형태로 설명하겠다.

- 항목 = 형태 (디폴트 값)

  범위 또는 크기

  내용

그러면, 위에서 설명한 환경파일의 각 절에는 어떠한 내용이 들어가는지 간단하게 살펴보겠다. 이 절에서는 우선, 간단한 환경파일 구성을 위한 기본적인 정보만을 제공할 것이다. 이 기본적인 환경파일 구성 외에 다른 사항들, 특히 특정 상황에 필요한 기능들이나 특수한 기능들은 앞으로 이어지는 장

에서 설명할 것이다. 따라서 본 장에서 설명할 내용은 JEUS Web Server 를 구동하기 위한 최소한의 정보이다.

JEUS Web Server 를 구동하여 기본적인 서비스를 하기 위하여 필수적으로 필요한 절은 DOMAIN, NODE, SVRGROUP, SERVER 절이다. 따라서 이 장에서는 이에 대한 간략한 설명을 할 것이다. 그리고 이어지는 장에서 간단한 환경파일 구성을 설명한다.

4.3 환경 파일의 구조

JEUS Web Server 의 환경파일에는 다음과 같은 4 개의 중요한 절이 있다.

- DOMAIN 절
- NODE 절
- SVRGROUP 절
- SERVER 절

4.3.1 DOMAIN 절

DOMAIN 절은 독립적인 JEUS Web Server 시스템 전체의 전반적인 환경에 대해 정의하는 부분이다. Domain 의 이름은 호스트 이름과 함께 암호화 되어 JEUS Web Server 의 License 확인에 사용된다.

DOMAIN 절에서 정의된 내용은 모든 Node 에 공통적으로 적용된다. 이 절에서 정의된 내용을 하위 절에서 재정의 할 수 있다. 시스템 전체적으로 기본적인 사항이 DOMAIN 절에서 정의되며, Node 에서 재정의된 항목은 해당 Node 에서 우선적으로 적용 된다.

이제부터 DOMAIN 절의 항목들에 대해 자세히 살펴보겠다.

DOMAIN 절의 기본 환경설정 형식은 다음과 같이 정의한다.

```
Domain Name       [,NliveInq]
```

필수항목

1) **Domain Name =** string

31 자 이내의 string 형식으로 정의한다. 31 자를 초과하는 이름을 사용할 경우, 컴파일 에러는 나지 않으나, 내부적으로 31 자까지만 인식하게 된

다. 31 자 이내로 이름을 설정하게 되어있는 기타 다른 절들의 항목에서도 이 점은 동일하다. Domain 이름은 호스트 이름과 함께 암호화 되어 시스템의 License 확인에 사용된다.

```
*DOMAIN
 JEUS_Web_Server1
```

위의 예는 도메인 절을 정의하고 있다. JEUS_Web_Server1 은 도메인의 이름으로, DOMAIN 절에서 정의되어야 하는 유일한 필수 항목이다.

4.3.2 NODE 절

NODE 절에는 다음과 같은 내용들이 정의될 수 있다.

- JEUS Web Server 시스템의 홈 디렉토리
- HTML 문서들이 들어있는 최상위 디렉토리
- 실행할 JEUS Web Server Process (HTH)의 개수
- 공유 메모리의 Key 값
- Port 번호 설정
- Listener IP Address 설정
- 기타 Logging 등...

JEUS Web Server 를 기동 시키면 Node 단위 별로 WSM, HTL, HTH Process 와 HTML, CGI 등의 실제 Service 를 수행하는 Server Process 들이 기동 된다. 이러한 Process 들이 기동하기 위해서는 실행 파일들이 존재하는 위치를 알아야 한다. 또한 JEUS Web Server Process 간 통신에 필요한 디렉토리와 각종 Error, 경고 메시지를 저장하는 디렉토리도 지정할 수 있다. 그리고 각 항목별로 지시자들의 형태가 고정되어 있다. 이는 보통 Literal, Numeric, String 등의 형태로 나타나는데, 이들의 형태에 맞게 각 항목을 설정하여야 한다.

NODE 절의 기본 환경설정 형식은 다음과 같이 정의한다:

```
[Node name]       WEBTOBDIR=”JEUS Web Server-home-path”,
                  SHMKEY =shared-memory-segment-key
                  DOCROOT =”JEUS Web Server-html-root”
                  [,User][,Group][,Admin][,HostName]
                  [,Method][,HTH][,HthQTimeout][,NodePort]
                  [,Port][,SslFlag][,JSVPort][,RacPort]
```

```
[,SslName][,MaxIdleTime][,CliChkIntval]
[,CacheSize][,CacheEntry][,KeepAliveTimeout]
[,KeepAliveMax][,AppDir][,PathDir][,SysLogDir]
[,UsrLogDir][,UserDir][,ErrorLog][,EnvFile]
[,IndexName][Listen][,DirIndex][,Options]
[,LanguagePrio][,Logging][,NodeType]
```

필수항목

1) **Node Name** = string

31 자 이내의 string 으로 정의한다. 31 자를 초과하는 이름을 사용 할 경우에는 컴파일 에러는 나지 않으나 내부적으로 31 자까지만 인식한다. Node 이름은 실제 등록된 Host 의 이름을 말한다. 예를 들어 UNIX 의 경우 "`uname -n`" 명령으로 각 Host 의 이름을 확인할 수 있다. 또한 해당 Node 명은 반드시 "`/etc/hosts`" 파일에 등록되어 있어야 한다. 하나의 Domain 은 하나 이상의 Node 로 이루어지므로, NODE 절에는 최소한 하나 이상의 Node 이름이 정의되어야 한다

2) **WEBTOBDIR** = literal

크기 : 255 자 이내.

JEUS Web Server 가 설치되어 있는 홈 디렉토리의 절대 경로명이다. 이 경로명은 환경변수 "JEUS_WSDIR"과 동일한 값이 정의 되어야 한다. JEUS Web Server 관련 작업은, 모두 JEUS Web Server 디렉토리 하에서 이루어진다.

이러한 bin, config, lib, docs, log, path 등의 하위 디렉토리 구조를 가지는 상위 디렉토리 명을 WEBTOBDIR 에 정의한다.

3) **SHMKEY** = numeric (32768 ~ 262143)

공유 메모리 세그먼트(Shared Memory Segment)를 가리키는 값이다. JEUS Web Server 를 이루는 Process 들이, 서로 정보를 공유하기 위한 공유 메모리 Key 값을 정의할 필요가 있다. 공유 메모리 Key 값을 정의하기 전에 이 Key 값들이, 다른 프로그램 또는 다른 업무에서 사용되는지 반드시 확인해야 한다. 그렇지 않으면 JEUS Web Server 가 Booting 시에 이 프로그램과 충돌을 일으켜서 실행이 되지 않는다.

현재 JEUS Web Server 에서 정의되는 Shared Memory 의 Key 값은 최소 32768 에서 최대 262143 까지 이다. 따라서, 이 범위 내에 있는 값을 이용하면 된다.

4) **DOCROOT** = literal

JEUS Web Server 가 Web 을 통해 Service 하는 모든 문서를 포함하는 루트 디렉토리의 경로이다. 즉, JEUS Web Server 는 DOCROOT 가 지정한 디렉토리를 최상위로 하여 문서를 Service 하게 된다. Client 가 요구한 URL 은 DOCROOT 의 경로 뒤에 추가되어 실제 경로명을 이루게 된다. JEUS Web Server 는 이 경로를 가지고 파일에 접근하게 된다.

선택항목

1) **HTH** = numeric (Default Number: 1, Max Number: 20)

JEUS Web Server 에서 가장 중요한 역할을 담당하고 있는 HTH (HTTP Request Handler) Process 의 개수를 설정한다. HTH 는 실질적으로 Client Browser 와 JEUS Web Server 내부 Service Process 사이를 중계하는 Process 이다. 즉, Client 의 요청을 받아 Service 를 받을 수 있도록 적당한 Process 에 넘겨주고, 다시 처리된 결과를 수신하여 Client 에게 되돌려 준다.

JEUS Web Server 에서는 모든 Client 가 이 HTH Process 에 연결되도록 설계되어 있다. 따라서 많은 수의 동시 접속자를 유지하기 위해서는 이 수를 적당히 늘려주는 것도 필요하다. 그러나 보통 하나의 HTH Process 가 적어도 800 개 이상의 Client 를 수용할 수 있기 때문에 하나의 HTH 로도 큰 문제는 되지 않을 것이다. 만약 1000 명 이상의 동시 사용자를 수용하여야 하는 시스템에서는, 두 개 이상의 HTH Process 를 구동하여야 한다. 현재 JEUS Web Server 에서는 최대 20 개까지의 HTH Process 를 구동하는 것이 가능하다. 따라서 약 16000 명의 사용자를 동시에 처리하는 것이 가능하다.

2) **PORT** = literal (Default Port: 80)

JEUS Web Server 의 HTTP Listener 포트 번호를 설정한다. 이는 기본적으로 Web Service 를 하기 위해서는 반드시 필요한 설정으로, 이 항목은 반드시 설정해야 한다. 보통 일반적인 Web Server 는 80 Port 를 이용한다. 그러므로 이 항목을 설정하지 않았을 경우에는 default 로 80 Port 가 설정된다. 그러나 다른 용도나 Internet 등의 경우에는 임의의 Port 를 설정하여 이용할 수 있다.

Virtual Host 를 사용하는 경우, Port 번호는 VHOST 절에서 재정의 해야 한다. 또한, JEUS Web Server 의 경우 여러 개의 Port 를 동시에 정의하는 것도 가능하다. 현재는 최대 10 개의 Port 를 동시에 설정하는 것이 가능하다.

즉, PORT 의 설정을 통해서 여러 개의 Port 가 동시에 Listen 하는 것이 가능하다는 것이다. 이에 대한 간단한 예는 다음과 같다.

```
PORT = “9090, 9091”
```

위와 같은 Setting 이 이루어지면, JEUS Web Server 는 9090 과 9091 Port 에서 모두 Web Service 를 받아들일 수 있다.

한가지 주의할 점은 이것은 Listen 항목과 동시에 운영할 수 없다는 것이다. Listen 항목에서 특정 Port 를 지정하는 경우 이 Port 만을 이용하게 된다. 즉, 중요한 것은 Listen 항목에서 지정된 Port 가 PORT 항목에서 지정된 것보다 우선된다는 것이다. 물론 Listen 항목에서도 여러 개의 Port 를 지정하는 것이 가능하다. 이에 대한 예는 Listen 항목의 설명에서 할 것이다.

3) **User, Group 설정 (UNIX 환경)**

JEUS Web Server 에서는 시스템의 보안을 위하여 JEUS Web Server 의 실제 실행 Process 에 대한 권한 설정을 할 수 있게 하였다. 만약 Web Server 의 실행 Process 들이 root 권한으로 실행되는 경우, 이에 문제가 발생하게 되면 시스템 전체에 큰 피해를 주게 될 수도 있다. 따라서, JEUS Web Server 에서는 이를 방지하고자 JEUS Web Server 의 실행 Process 들을 root 권한이 아닌 다른 권한으로 실행할 수 있게 하였다.

그러나 이러한 실행 권한의 변환은 root 만이 가능하기 때문에 이를 설정하기 위해서는 반드시 root 로 작업을 하여야 한다. 또한, 자신이 설정하고자 하는 user 와 group 이 시스템에 설정되어 있어야 한다. 이에 대한 사항은 자신이 운영하는 운영체제의 Manual 을 참조하기 바란다.

4) **User** = literal

설정된 User 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

5) **Group** = literal

설정된 Group 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 Group 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

6) **Admin** = literal

관리자의 정보를 나타낸다. 여기에는 관리자에게 연락할 수 있는 E-mail 주소를 설정할 수 있다.

7) **HostName** = literal

이 항목을 설정하면 Http Response Header 의 host name Field 에 기록을 남겨준다. JEUS Web Server 가 설치된 machine 의 Domain name 을 "www.tmax.co.kr"과 같이 넣어주면 된다.

8) **JSVPort** = numeric (Default Port: 9999)

JEUS Web Server 와 Java Servlet 수행 Server 간의 연결 Port 번호이다 .

9) **CheckIntval** = numeric (Default Number: 30 second)

JEUS Web Server 에서 접속한 Client 가 살았나 죽었나를 확인할 필요가 있을 때 설정하는 항목이다. 항목 시간 간격을 설정함으로써 JEUS Web Server 는 Client 의 생존 여부를 주어진 시간 간격마다 확인하고 죽었을 경우 JEUS Web Server 쪽에서 접속을 끊는다.

10) **CacheSize** = numeric (Default Size: 128 Kbyte)

JEUS Web Server 는 Server 내부 Caching 기능을 지원한다. 따라서 많은 Web Application 이 용이하게 처리되어 질 수 있다. 예를 들어 세션 정보를 관리해야 하는 경우 기존의 방식처럼 쿠키를 쓰지 않고 JEUS Web Server 가 내부적으로 제공하고 있는 Cache 에 정보를 저장함으로써 더 편리하고 효율적인 작업을 수행할 수 있다. 이러한 내부 Cache 의 사이즈를 필요에 따라 조정할 수 있다. 여기서는 Cache 의 한 엔트리의 크기로서 기본단위는 Kbyte 이다.

11) **CacheEntry** = numeric (Default Number: 128)

Cache 의 총 Hashing Key 엔트리 개수를 설정한다. JEUS Web Server 에서는 Hashing 방식을 이용하기 때문에 이의 값에 따라서 Cache 기능의 성능이 영향을 받게 된다. 만약 이 값이 적게 설정되면 Hashing Key 값이 적게

되어 Key 는 쉽게 찾지만, 각 Key 에서의 값을 찾는 문제가 발생하고 Key 값이 많게 되면 다양한 Key 에 대한 값이 나오지만 각 Entry 가 적게 되어 쉽게 찾을 수 있다. JEUS Web Server 의 기본 설정 값은 128 이다. 이 값을 변경하여도 큰 영향은 없지만 가급적 128 개의 Hashing 값을 권장하는 바이다..

12) **KeepAliveMax** = numeric (Default Number : 9999)

보통은 하나의 Client 가 한 개 이상의 요구를 연속적으로 Server 에 요청하는 경우가 많다. 이러한 경우 매 요구마다 연결을 다시 맺어야 한다면 비효율적일 것이다. 따라서 일정 개수의 요구는 처음 커넥션을 유지한 상태로 Service 를 하고 커넥션을 끊도록 한다. 커넥션을 끊기 전에 들어주는 요구의 개수를 KeepAliveMax 에서 지정한다

13) **KeepAliveTimeout** = numeric (Default Number : 60)

하나의 Client 가 불필요하게 커넥션을 오래 잡고 있는 경우를 막기 위해서 요구간, 시간 간격이 일정 시간 이상이 되면 커넥션을 끊을 수 있도록 설정할 수 있다.

이러한 요구간 시간 간격은 KeepAliveTimeout 값으로 설정한다. Default 로 설정되는 값은 60 이며 단위는 second 이다.

14) **Timeout** = numeric (Default Number : 300)

사용자가 접속을 하여 Data 를 내려 받거나, 사용자의 요구를 내려 받는 시간을 지정하는 것이 가능하다. 이 때 이 Timeout Field 를 이용하게 되는데, 이 Field 를 통해서 사용자의 최대 접속시간을 지정할 수 있다. 이는 사용자와 맺은 연결이 문제가 생겨 계속해서 의미 없는 Data 전송이 발생할 때 이를 방지하기 위해서 이용한다. Default 로 설정되는 값은 300 이며 단위는 second 이다..

15) **UserDir** = literal

JEUS Web Server 를 통해 여러 사용자를 동시에 서비스 하려는 경우 필요하다. 이 때 들어가는 값은 각 사용자의 디렉토리의 이름이다. 이를 설정하면 JEUS Web Server 는 각 사용자의 디렉토리를 찾아서 서비스를 시작한다.

16) **AppDir** = literal

JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 디렉토리의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

17) 기본 Log 디렉토리 설정

JEUS Web Server 에서는 기본적으로 Log 정보를 남기기 위하여 반드시 설정하여야 하는 디렉토리들이 존재한다. 이는 JEUS Web Server 에 장애가 발생하거나 혹은 JEUS Web Server 로 특정 Service 를 하였을 때 문제가 생기는 경우, 이를 처리하기 위하여 이에 대한 기록을 남기는 것이다.

이는 환경파일에서 따로 설정하지 않아도 기본적으로 WEBTOBDIR 로 설정된 디렉토리밑에 log 디렉토리에 만들어지게 되는데, 주의할 점은 JEUS Web Server 의 Booting 전에 반드시 이에 대한 Directory 가 실제로 존재하는지를 확인하여야 한다는 것이다. 만약 디렉토리가 없다면 JEUS Web Server booting 시에 이를 찾다가 파일을 찾지 못한다는 에러를 리턴하면서 booting 이 되지 않는다. 그러므로 어떤 오류로 인하여 log Directory 가 만들어지지 않았다면 반드시 Log 디렉토리를 먼저 만들어 주고 JEUS Web Server 를 실행하여야 한다.

반드시 설정하여야 하는 디렉토리에는 트랜젝션 관련 로그를 기록하는 디렉토리(default : (WEBTOBDIR)/log/txlog)와 시스템 관련 로그를 기록하는 디렉토리(default : (WEBTOBDIR)/log/syslog)가 있는데 이들 디렉토리들은 Install script 를 이용하여 JEUS Web Server 를 설치하였다면 default 경로로 자동으로 생성된다. 이 경로는 사용자가 다시 재정의하여 사용할 수 있는데 그 방법은 아래 설명하는 SysLogDir 이라는 항목을 설정하면 된다.

18) **SysLogDir** = literal (Default Path : (WEBTOBDIR)/log/syslog)

시스템 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다. 시스템 메시지란 wsm, htl, hth 등 JEUS Web Server 기동의 핵심 Process 들이 발생한 메시지들과 시스템 내부적으로 발생한 메시지들을 일컫는다. 이 항목은 오직 SysLogDir 를 재정의하여 사용하기 위한 것으로, 이 항목을 설정하지 않는 경우에는 Default Path 에 Log 가 남는다. Default Path 를 수정하여 사용할 경우 주의를 요한다.

19) **UsrLogDir** = literal (Default Path : (WEBTOBDIR)/log/usrlog)

사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

20) **IndexName** = literal (Default : index.html)

Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때 기본적으로 Service 되는 파일 이름을 설정한다. 따로 설정하지 않으면 `index.html` 이 설정된다.

21) **Options** = literal

Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때의 동작을 지정한다. 보통 사용자가 특정 URI 를 보내고 디렉토리 이름만 요구 하였다면 사용자에게 Directory 의 내용을 보여주는 것이 가능하다. 물론 원하지 않는 다면 보여 주지 않을 수 있다. 기본적인 설정은 보여주지 않는 것으로 되어 있다. 이 때 '+' 나 '–' Option 들을 이용하는데, 만약 디렉토리 정보를 보여주길 원한다면 "+Index"와 같이 설정하면 된다..

이외에도 그 Node 에서 CGI 를 쓰고 싶지 않은 경우 –CGI 와 같이 설정하면 CGI 가 수행되지 않게 된다. 즉 + 나 – Option 을 통해서 많은 Service 들 혹은 기능들의 수행여부를 결정할 수 있다는 것이다. 이에 이 부분에서 설정 가능한 Service 들과 기능들은 다음과 같다. 만약 ALL 을 기입하면 모든 기능에 대한 설정을 하는 것이다. 즉, -ALL 은 모든 기능을 이용하지 않겠다는 것이고 , +ALL 은 모든 기능을 이용하겠다는 것이다..

Service: HTML, CGI, SSI, PHP, JSV

기능: INDEX, INCLUDE

22) **Method** = literal

Client 가 보내는 Request Method 에 대한 정의를 할 수 있다. 이 때 기본적으로 GET, POST, HEAD 등이 있어 이를 기본적으로 지원하고 만약 이들 중 특정 Method 를 쓰고 싶지 않은 경우 – Option 을 이용하여 제거할 수 있다.

23) **Listen** = literal

JEUS Web Server 가 Booting 될 때, 원하는 IP Address 에서 연결을 맺도록 할 수 있다. 즉, 여러 개의 IP Address 를 가진 Server 에서 자신이 원하는

IP Address 에서만 Service 를 원하는 경우, 이 값을 정할 수 있다. 이 때 여러 개를 복수로 설정하는 것도 가능하다.

24) **DirIndex** = literal

뒤의 DIRINDEX 절에서 설정하는 디렉토리 인덱스의 이름을 적어준다.

25) **LanguagePrio** = literal

접속 Client 가 사용 언어를 지정하지 않았을 경우 Server 쪽에서 지정된 언어 순서대로 Multiview request 등의 처리가 이루어지도록 한다.

26) **Logging** = literal

뒤의 Logging 절에서 설정하는 Logging Name 을 써준다. 이 이름을 가지고 이 Node 에서 그에 해당하는 Log 를 남기게 되는 것이다.

27) **NodeType** = string

JEUS Web Server 는 Node 단위로 Server 에 특정한 역할을 부여할 수 있다. 예를 들어 하나의 Node 를 프락시 Server(Proxy Server)로 쓰고 싶다면 NodeType 에서 Proxy 로 설정하면 된다.

28) **ErrorLog** = literal

오류 발생시 설정하는 Logging 정보 이름을 써준다. 이 이름 또한 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다.

29) **EnvFile** = string

JEUS Web Server 에서 특정 정보를 읽어 들일 필요가 있는 경우 이용된다. 즉, 어떤 파일에 변수와 변수에 대한 정보를 기록한 후, JEUS Web Server 기동 시에 이 정보를 읽어야 하는 경우 이 EnvFile 에 등록된 파일을 읽어 들인다. 즉, JEUS Web Server 가 기동 시에 이 파일을 읽어서 그 정보를 가지고 있게 된다..

30) **SSLFLAG** = string (Y | N) (default : N)

JEUS Web Server 에서 SSL 을 이용할 때 반드시 지정하여야 하는 항목이다. 이 SSLFLAG 가 Y 상태이면, 그 Node 에서 SSL 을 이용하겠다는 것이고, N 상태이면, 이용하지 않는다는 것이다. 기본 설정은 SSL 을 이용하지 않는 것으로 되어 있다.

31) **SSLNAME** = string

JEUS Web Server 에서 SSL 을 이용하는 경우, 이에 대한 설정을 나타내는 것이다. 이는 반드시 SSLFLAG 가 on 이 된 상태에서 적용되어야 하며 off 나 설정이 되지 않은 상황에서는 아무런 의미가 없다. 이 때 지정되는 이름은 반드시 SSL 절에서 선언이 되어 있어야 한다.

4.3.3 SVRGROUP 절

JEUS Web Server 를 통해 응용 Server Process 를 접근하는 경우 Server Process 의 논리적인 연관성에 따라 이들을 그룹으로 관리할 필요가 있게 된다. 이 절에서는 이러한 그룹에 대한 환경 설정이 이루어 진다. Node 이름, Server 의 종류, 호스트의 이름 등을 등록한다.

이 밖에 NODE 절이나 VHOST 절에서 정의한 내용이 Server Group 에 따라 새롭게 정의할 수 있으며 데이터베이스를 사용하는 경우 데이터 베이스 접근과 관련된 정보들이 정의될 수 있다.

SVRGROUP 절에는 기본적으로 다음과 같은 내용이 정의될 수 있다:

```
SVRGROUP Name      NODENAME,SVRTYPE
                   [,VhostName][,Cousin][,Backup][,Load]
                   [,AppDir][,UsrLogDir][,AuthentName][,DBName]
                   [,OpenInfo][,CloseInfo][,MinTms][,MaxTms]
                   [,LOGGING][,EnvFile]
```

필수항목

Server Group name = string

크기 : 31 자 이하

Server Group 에 대한 논리적인 이름으로서 SVRGROUP 절 내에서 유일한 값이어야 한다. SVRGROUP 절 이름은 SERVER 절의 SVGNAME 항목에서 사용된다.

NODENAME = string

Server Group 이 존재하는 Node 를 정의한다. 사용되는 NODENAME 은 NODE 절에서 정의한 Node 이름이어야 하며, Node 이름은 유닉스 명령어 "`uname -n`"을 이용해서 확인해 볼 수 있다.

SVRTYPE = string

Server Group 의 속성, 즉 어떠한 Service 를 제공하는가를 명시한다. Server 타입으로 HTML, CGI, JSV, WEBSTD, TPSTD, SSI 등을 명시할 수 있다.

4.3.4 SERVER 절

SERVER 절에서는 실질적으로 제공하는 Service 들을 등록한다. JEUS Web Server 는 등록된 Service 만을 처리하기 때문에 새로운 Server 프로그램이 추가되는 경우 Server 절의 환경파일에 반드시 등록하여야 한다. JEUS Web Server 가 제공하는 대부분의 Service 는 SERVER 절에서 등록이 가능하며 비즈니스 로직을 JEUS Web Server 를 통해 직접 호출하는 경우에만 SERVICE 절의 등록이 필요하다. 각각의 Server 는 위의 Server Group 절에 정의된 Service 종류에 따라 HTML, CGI, JSV 등으로 구분되며 Server Group 이름과 Process 의 가능한 개수 등이 같이 등록된다.

SERVER 절에는 다음과 같은 내용이 정의된다:

- 각 Server Process 가 속하는 Server Group.
- Server Process 의 최소 개수와 최대 개수.

SERVER 절의 기본 형식은 다음과 같다:

```
ServerName        SVGNAME=server-group-name,
                  [,Clopt][,SeqNo][,MinProc][,MaxProc]
                  [,UsrLogDir][,UriDir][,MaxQCount]
                  [,ASQCount][,MaxRestart][,SvrCPC]
                  [,SVRTYPE]
```

필수항목

Server Name = string

크기 : 31 자 이하

Server 의 실행 파일 이름으로서 일반적으로 Server 이름은 유일(Unique) 해야 한다. 즉 하나의 Server 이름은 SERVER 절에 단 한번만 정의되어야 한다.

같은 이름을 중복하여 이용하면 환경 파일의 Compile 시에 Error 가 발생하게 된다.

SVGNAME = string

Server 가 속해 있는 Server Group 을 정의한다.

여기에 사용되는 값은 반드시 SVRGROUP 절에서 정의된 Server Group 이름이어야 한다. Server 와 SVRGROUP 절의 연결을 통해서 Server 가 어떤 Node 에서 동작할 것인지, 어떤 리소스 매니저(데이터 베이스)를 사용하는지 알 수 있으며. 해당 리소스 매니저를 열 때 필요한 파라미터를 넘겨 줄 수 있다.

4.4 JEUS Web Server 환경파일 작성

4.4.1 소개

이 장에서는 JEUS Web Server 를 위한 간단한 환경 파일을 만드는 방법에 대해서 알아볼 것이다. 이 장에서는 JEUS Web Server 를 기동하기 위한 최소한의 내용을 설명할 것이고, 실제 환경 파일을 만드는 과정에 대해서도 설명할 것이다.

4.4.2 기본 설정

JEUS Web Server 가 제공하는 모든 기능을 이용하기 위해서는 환경 파일의 모든 내용을 파악하는 것이 좋다. 따라서 앞 장에서 제공하는 JEUS Web Server 환경 파일의 설명을 탐독하여 JEUS Web Server 의 모든 기능을 파악하면 JEUS Web Server 에서 제공하는 기능을 모두 이용할 수 있다.

JEUS Web Server 에서 제공하는 최소한의 환경 파일은 DOMAIN 절, NODE 절, SVRGROUP 절, SERVER 절만이 필요하다. 그럼 이들을 가지고 최소한의 환경 파일을 만들어 보자. 예제로 사용된 환경 파일은 JEUS Web Server 에서 HTML, CGI, SSI 등의 기본적인 Service 등을 이용하게 해 주는 최소한의 것이다. 그럼 하나 하나 설정된 값에 대해 설명하겠다.

(보통 환경 파일은 JEUS Web Server 의 `config/` Directory 에 저장한다.)

DOMAIN Section 설정하기

먼저, DOMAIN 절을 살펴 보자. DOMAIN 절은 간단하게 DOMAIN name 만을 설정하면 된다. 이 외에도 추가 기능들이 있으나, DOMAIN 절에서는 이것만으로 충분히 Service 가 가능하다.

NODE Section 설정하기

이 절에서는 JEUS Web Server 가 Booting 시에 기본적으로 가져야 하는 정보들에 대한 정의가 이루어진다. NODE 절에서는 WEBTOBDIR 과 Shared Memory 를 위한 SHMKEY 값 그리고 `index.html` 등의 HTML 문서가 존재하는 디렉토리를 설정한 DOCROOT 그리고 마지막으로 JEUS Web Server 가 Listen 하고 있는 Listen PORT 를 설정한 PORT 이다. 아래 예제는 JEUS Web Server 가 8080 PORT 로 Listen 하고 있는 환경이다.

SVRGROUP Section 설정하기

이는 앞장에서 설명한 대로 JEUS Web Server 에서 제공하는 Server Type 별로 각 Group 을 설정한 것이다. 아래 예에서는 HTML 문서를 처리하기 위한 것으로 HTML Server Group 을 설정하고 이의 이름을 htmlg 를 설정하였다. 마찬가지로 CGI Server Group 으로 cgig 를 설정하였고, SSI Server Group 으로 ssig 를 설정하였다. 위의 SVRTYPE 에 정의된 것이 실제 Service 를 정의하는 것인데, JEUS Web Server 에서는 현재 HTML, CGI, SSI 등을 제공한다. 이와 같이 Group 을 설정하면, 이는 이어서 등장하는 SERVER 절에서 이를 가지고 실제 Server 를 운영할 수 있게 된다. 즉, SERVER 절에서는 SVRGROUP 에서 설정된 하나의 Server Group 을 가지고 여러 개의 Server 를 만드는 것이 가능하게 된다.

SERVER Section 설정하기

이는 JEUS Web Server 가 실제로 Service 를 처리하기 위한 것으로 아래 예에서는 html, cgi, ssi 등에 대한 것이 정의 되어 있다. 아래 환경 파일에서 html 을 예로 들어 설명하겠다.

```
html      SVGNAME = htmlg, MinProc = 1, MaxProc = 5
```

아래 환경 파일에서 위와 같이 설정되어 있는데, 이를 보면, html 이라는 Server 가 htmlg 라는 Server Group 에 속하여 있음을 알 수 있다. 그런데 위의 SVRGROUP 절에서 htmlg 는 HTML 이라는 Server Type 을 가진다고 설정이 되어 있다. 따라서, 이 html 이라는 Server 는 HTML Service 를 위한 것이라는 것을 알 수 있다. SSI 나 CGI 역시 이와 같은 원리로 이루어 진다. 그리고 뒤에 있는 MinProc 와 MaxProc 는 실제 Service 할 수 있는 Process 의 숫자이다.

4.4.3 환경 파일 예제

<<*http.m*>>

```
*DOMAIN
JEUS_Web_Server1
*NODE
webmain     WEBTOBDIR = "/usr/local/webserver",
            SHMKEY = 69000,
            DOCROOT = "/usr/local/webserver/docs",
            PORT = "8080"
*SVRGROUP
htmlg       NODENAME = webmain, SVRTYPE = HTML
cgig        NODENAME = webmain, SVRTYPE = CGI
ssig        NODENAME = webmain, SVRTYPE = SSI
```

```
*SERVER
html        SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi         SVGNAME = cgig, MinProc = 1, MaxProc = 5
ssi         SVGNAME = ssig, MinProc = 1, MaxProc = 5
```

4.5 동적 컨텐츠를 위한 JEUS Web Server 설정

4.5.1 소개

본 장에서는 앞에서 살펴본 기본적인 설정 외에 특수한 작업을 하기 위한 설정에 대하여 알아 볼 것이다. 이제 Web Server 가 설치되었으므로 html 을 이용한 서비스는 가능하게 되었다. 즉, JEUS Web Server 의 Document Directory 에 html 문서를 두면 브라우저로 호출하여 열람이 가능하다. 이제 좀더 고급스러운 서비스를 위한 동적 서비스의 설정을 알아볼 차례다. 또한 가상 호스트를 이용한 다중서버 서비스의 구축 방법도 알아보자.

본 장을 통해 CGI, SSI, PHP 를 사용하기 위한 설정과 간단한 예제를 통해 그 설치 예를 보여주고자 한다. 웹을 풍요롭게 하는 이러한 Tool 들의 자세한 작성법은 본 매뉴얼에서 다룰 내용은 아니지만, 각각의 특징을 안다면 웹의 설계에 도움이 된다. 어떤 서비스를 무엇을 사용해 구현하는 것이 효과적일지 안다면 이후 그에 대한 전문 지식은 다른 서적을 참고하면 될 것이다. 이해를 돕기 위해 각각에 대한 기본적인 설명과 예제를 담았다.

4.5.2 CGI 설정

CGI 의 이용

CGI 는 Common Gateway Interface 의 약자이다. WWW 에서는 HTML 에 의해 여러 가지 정보를 처리한다. HTML 은 홈페이지를 만드는 기초가 되는 언어이다. 하지만 HTML 만으로 모든 정보를 다 처리할 수 없다. 단 방향식의 정보제공 역할만 할 따름이다. 이것을 보충하기 위해서 여러 가지 방법이 마련되어왔다. 그 중 하나가 외부 프로그램을 수행하여 그것의 결과를 HTML 형태로 보여주는 방식이다. 이때 이 외부프로그램과 Web Server 간의 연결 역할을 하는 것이 CGI 이다.

또는 넓은 의미로 CGI 를 수행하는 외부 프로그램을 포함하여 말하기도 한다. 예를 들어, 홈페이지에 방문객들의 comment 를 받을 수 있는 방명록을 만들려고 할 때, 웹에서 구현하는 HTML 만으로는 해결할 수 없다. 그래서 외부 프로그램이 필요한데, 이 때 외부 프로그램과 Web Server 간에 서로 주고받을 수 있는 규약을 CGI 라고 하고, 그 때 사용하는 프로그램을 흔히 CGI 프

로그램(혹은 CGI 스크립트) 이라고 한다. 이 CGI 프로그램은 통상적으로 C/C++ 나 PERL 혹은 UNIX Shell, Tcl/Tk 등을 사용하여 구현한다.

홈페이지를 interative 한 형태로 만들 수 있는 CGI 프로그램의 종류는 매우 다양하다. 방문객 카운터나 방명록 뿐만 아니라 웹 게시판, 웹 대화방, 검색 엔진, 다양한 배너 보여주기, 업로드가 가능한 자료실, 폼을 이용하여 메일을 띄우는 폼 메일(Form Mail) 등 이루 헤아릴 수 없이 많다. 현재 이러한 CGI 들은 표준화 되어 있다고 봐도 된다. 또한, 이것은 외부 프로그램이 수행되는 방식이기 때문에, 프로그램 자체에 문제가 없다면 다른 곳에 서로 이식하는 데에도 아무 문제가 없다. 즉, 어떤 Web Server 에서 무리 없이 돌아가는 CGI 라면 다른 Web Server 에 거의 무난하게 이식이 가능하다는 것이다. JEUS Web Server 역시 다른 Web Server 와 같은 방식으로 처리하기 때문에 CGI 수행에는 전혀 지장이 없다.

CGI 를 제작하는 방법은 매우 다양하고, 여러 가지 방법이 있으므로 본서에서 깊은 내용을 언급하지는 않을 것이다. CGI 프로그램을 제작하는 것은 시중에 책으로 많이 나와 있고, 또한, 온라인 상에서도 많은 자료를 구할 수 있다.

CGI 예

부록 D 를 참조.

4.5.3 SSI 설정

SSI 의 이용

SSI 는 다이나믹한 문서를 만드는데 있어 매우 유용하다. 예를 들면 헤더 파일 등을 더할 수도 있고 마지막 수정일(Last Modified)을 자동으로 조절할 수 있는 기능들을 문서에 포함시킬 수도 있다. 이것은 CGI 와 같이 사용할 수 있으나 복잡하지 않으며, 프로그래밍 또는 스크립트의 기능 같은 것은 가지고 있지 않다. SSI 는 간단하게나마 다이나믹한 문서를 만들 수 있는 것이다.

SSI 는 HTML 문서에 'command' 를 집어 넣어 사용할 수 있으며, Server 에서는 SSI 문서를 읽어 들이고, SSI 명령어를 찾아보고 그에 맞는 기능을 수행한다. 예를 들면 Last modification time 을 수정하는 SSI 명령어가 문서에 포함되어 있으면 서버는 파일로부터 명령어들을 읽어 들여 명령을 수행하여 시간을 수정한다.

SSI 는 참고 자료를 구하는 것이 쉽지가 않다. 대부분 CGI 프로그램 작성란에 참고로 들어가거나, 혹은 간단한 예제로 넘기는 경우가 대부분이다. 따라서 여기에서 간단하나마 기본적인 SSI 의 설정과 문법을 알아 보도록 하자.

JEUS Web Server 에서 기본 설정은 HTML 파일 안에 SSI 명령어를 포함하고 있지 않다. 왜냐하면, HTML 파일을 매일 액세스 하는 곳에서는 이로 인하여 html 이 오히려 느리게 작동하기 때문이다. SSI 가 필요해서 사용하기 원한다면 JEUS Web Server 환경 파일 안에 SSI 를 위한 Server 를 추가로 포함시켜 주어야 한다. 이렇게 하는 것이 성능 향상에 더 유리한 구조이다.

<<*http.m*>>

```
*SVRGROUP
ssig  NODENAME = webmain, SVRTYPE = SSI
*SERVER
ssi1  SVGNAME = ssig, MinProc = 1, MaxProc = 10
```

SSI Commands

모든 SSI 명령어는 HTML 문서 안에 HTML comments 형식으로 저장되어야 한다. SSI 사용방법은 아래와 같다:

```
<!--#flastmod file="nextel.html" -->
```

위 예에서 사용된 flastmod 명령어는 수정시간을 출력하라는 뜻이고 Value 로는 nextel.html 이 쓰인다.

명령어 전체는 comment <!--와 -->로 처리되어야 한다.

일반적으로, 모든 명령어는 다음 형식을 따른다:

```
<!--#command arg1="value1 arg2="value2 ... -->
```

여기서 arg1, arg2 는 인수고 나머지 value1, value2 는 인수의 값을 나타낸다. flastmod 예에서, 'file'은 인수가 되는 것이고 'nextel.html' 은 값이 되는 것이다. 흔히 명령어는 인수이름에 따라 다르게 실행되어 질 수 있다.

예)

```
<!--#flastmod virtual="/" -->
```

서버의 홈페이지 마지막 수정시간을 얻을 수 있다. (이것은 다른 파일명을 가지고 Access 될 때 유용하게 사용할 수 있다.)

SSI 명령어가 실행될 때 'environment variables' 값이 설정이 된다. 이것은 CGI variables 을 포함하고 있고(REMOTE_HOST etc), DOCUMENT_NAME 그리고 LAST_MODIFIED 등을 가지고 있다. 또 다른 예로 echo 명령어로도 출력할 수 있다.

```
<!--#echo var="LAST_MODIFIED" -->
```

SSI 예

부록 D 를 참조.

4.5.4 PHP Configuration

PHP 는 Perl 과 유사한 형태의 Script 로 간편성과 괜찮은 성능으로 인하여 많이 이용되고 있다. 속도, 개발 편의성, 여러 가지 확장 기능으로 볼 때 기존의 펄(Perl) 보다 한 수 위인 언어로 LINUX 나 UNIX 계열 뿐만 아니라, WIN32 용 바이너리 파일을 제공해 Microsoft 계열의 Web Server 에서도 사용이 가능하기 때문에, 운영체제에 독립적인 웹 프로그램 개발이 가능한 것이 큰 장점이다. 가장 기본적인 레벨에서, PHP 는 CGI 프로그램에서 할 수 있는 모든 것을 할 수 있다. form data 를 가져오고, 동적인 웹 페이지를 만들거나, Cookie 를 보내고 받을 수도 있다.

PHP 의 이용

JEUS Web Server 에서 PHP 를 이용하기 위해서는 약간의 설치 작업이 필요하다. 이는 HTML 이나 CGI 를 이용하는 것과 거의 유사하기 때문에 쉽게 적용할 수 있을 것이다.

먼저, SVRGROUP 절에 PHP 에 관련된 Group 을 설정하고 이를 Server 절에서 다시 정의하면 된다. 이 때, SVRGROUP 에 ScriptLoc 라는 항목이 추가 되는데, 이는 php 의 실제 실행 모듈이 있는 곳을 말한다. 이는 보통 `cgi-bin/` 디렉토리에 추가하는데 JEUS Web Server 에서도 이를 권장한다. 아래는 이에 대한 간단한 예이다:

PHP 모듈이 있는 곳의 경로는 WEBTOBDIR 이하의 상대 경로 위치만 적어주면 된다. 즉, 아래 예에서 ScriptLoc 에 설정된 모듈경로는 절대 경로가 아니라, JEUS Web Server 가 설치된 디렉토리 아래에 `/cgi-bin/php` 가 위치한다는 의미이다.

<<*http.m*>>

```
*SVRGROUP
phpg  NODENAME = webmain, ScriptLoc = "/cgi-bin/php",
      SVRTYPE = PHP
*SERVER
php   SVGNAME = phpg, MinProc = 1, MaxProc = 10
```

위와 같이 설정을 하면 PHP 를 사용할 수 있다.

php3 모듈을 사용하는 경우에는 부가적인 설정이 필요하다. 다른 버전과는 달리 php3 의 경우는, 확장자가 .php3 인 파일을 이용하기 위해서 이를 EXT 절에 추가하여 주어야 한다. php4 는 기본적으로 php 라는 확장자를 이용하기 때문에 추가 설정을 요하지 않으나, php3 는 확장자가 .php3 이기 때문에 이는 EXT 절에 추가해 주어야 동작한다. 아래 예를 참조하자.

```
*EXT
php3  MimeType = "application/x-httpd-php3", SVRTYPE = PHP
```

PHP 예

부록 D 를 참조.

4.6 Virtual Hosting 설정

가상 호스트는 현재 http1.1 을 지원하는 브라우저에서 적용할 수 있는 Web Server 의 기능으로 하나의 Web Server 를 이용하여 마치 여러 대의 Web Server 가 운영되고 있는 것과 동일한 효과를 내는 것이다. JEUS Web Server 에서는 이 가상 호스팅 기능을 제공한다. 간단한 예를 들어 이 기능을 살펴보자.

당신이 웹 상에서 새로운 서비스를 시작하고, 그 서비스는 인터넷 신문사인 "JEUS Web Server Times" 라고 가정해 보자. 이 서비스를 하나의 IP 주소와 도메인 이름, 그리고 한대의 Web Server 를 이용해서 운영할 수도 있다. 그러나, 당신은 하나의 IP 를 사용하면서도 몇 개의 도메인 이름으로 분리하여 서비스를 제공한다면 이용자도 알기 쉽고 관리 및 차후 확장에도 용이할 것이라는 판단 하에 다음과 같은 몇 개의 서비스로 분리하고자 한다.

- webtimes.com: 메인 페이지
- society.webtimes.com: 사회면 기사를 다루는 섹션
- sports.webtimes.com: 스포츠 기사를 다루는 섹션

위와 같이 분리하여 HTML 문서도 만들고 기타 서비스도 제공한다. 위의 주소를 입력하여 접근할 수도 있는 것이다. (실제로 많은 신문사 사이트를 방문해 보면 위와 같은 구성으로 되어 있음을 확인할 수 있다.)

Web Server 에서는 위와 같은 설정을 적용하기 위해 두개의 Virtual Host 를 할당하여 각각 society 와 sports 에 적용해 주면 된다. 이렇게 하면 이들 서비스는 메인 페이지와는 다른 Web Server 에서 운영되는 것과 같은 효과를 줄

수 있으며, 실제 Web Server 내에서도 document 문서의 경로 및 기타 모든 설정들을 분리하여 사용할 수 있다. 이렇게 도메인 이름을 분리하여 하나의 IP 로 서비스 하는 방식을 Name Based Virtual Host 라 한다. 이와 구별되는 방식으로는 IP Address Based Virtual Host 가 있는데 이 방식은 다른 IP 를 사용하므로 사실상 별개의 도메인이라 할 수 있어, 그 설정에 있어서도 다른 IP 만 사용하면 되고 또 크게 쓰이지 않는 방식이다. (왜냐하면 한 사이트가 다량의 IP 를 확보하기는 힘들기 때문이다.)

가상 호스트의 구조를 그림으로 나타내어 보면 아래와 같다:

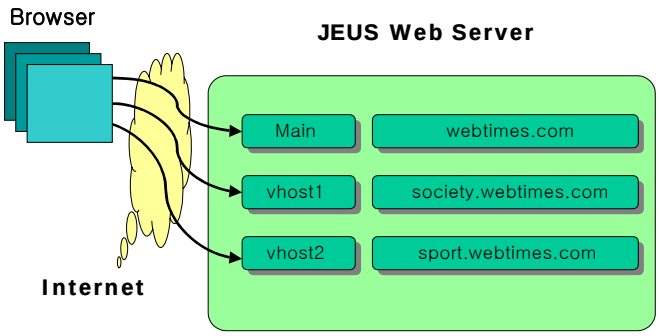

그림 4. 가상 호스트의 구조

위와 같이 하나의 IP 와 한대의 Web Server 를 이용해 각 도메인의 서비스를 할 수 있으며, 동일 IP 에 다른 이름으로 호출된 서비스를 JEUS Web Server 의 가상 호스트가 구분하여 서비스하게 된다.

아래의 예는 위의 Web Times 란 가상의 사이트를 바탕으로 구축한 가상 호스트의 예제이다.

<<http.m>>

```
*VHOST
vhost1 DOCROOT = "/usr/local/webserver/docs/vhost1_docs",
      NODENAME = webmain,
      HOSTNAME = "society.webtimes.com",
      PORT = "8090",
      User = "user1",
      Group = "group1",
      UsrLogDir = “/usr/local/webserver/vhost1/log/usrlog",
      IconDir = “/usr/local/webserver/vhost1/icons",
      UserDir = “/usr/loccal/webserver/vhost1",
```

```
      EnvFile = vhost1_env,
      IndexName = "v1_index1.html",
      LOGGING = "v1_log1",
      ERRORLOG = "v1_log2"

vhost2 DOCROOT = "/usr/local/webserver/docs/vhost2_docs",
      NODENAME = webmain,
      HOSTNAME = "sports.webtimes.com",
      PORT = "8092",
      User = "user2",
      Group = "group2",
      UsrLogDir = “/usr/loccal/webserver/vhost2/log/usrlog”,
      IconDir = “/usr/loccal/webserver/vhost2/icons",
      UserDir = “/usr/loccal/webserver/vhost2",
      EnvFile = vhost2_env,
      IndexName = "v2_index.html",
      LOGGING = "v2_log1",
      ERRORLOG = "v2_log2"
```

위와 같이 VHOST 절을 적절히 이용해 주면 효과적인 웹 운영이 가능하다.

4.7 Logging 설정

JEUS Web Server 는 Web Service 에 대한 요청과 그에 대한 제공 등 자신과 관련된 기록을 모두 환경 파일에 지정된 Log File 에 저장한다. 따라서 Log File 을 읽어보면 누가 언제 무엇을 요청했고 또 무엇을 가져갔는지 알 수 있으며 Web Server 에 얼마나 많은 사람이 왔는지 가장 오래 보는 또는 가장 많이 보는 페이지는 무엇인지 등을 알 수 있다. 이러한 Log File 은 위에서 말한 대로 환경 파일에 명시된 곳에 저장되며 이는 Web Server 를 운영할 때 임의의 위치에 저장이 가능하다. 또한 JEUS Web Server 는 한 개가 아닌 여러 개의 Log File 을 생성할 수도 있는데, 이는 기본적인 Access Log File 과 Error 에 관련된 Error Log File 등으로 나누어진다. 단, Log 를 남기는 과정에서 시스템에 약간의 부하가 발생한다. 이는 System 에 관련된 것과 Capacity 에 관련된 것이다.

먼저 System 에 관련된 것은 다음과 같다. JEUS Web Server 가 Log 를 남기는 과정에서 Disk 에 Writing 을 해야 하기 때문에 Log 를 남기지 않는 것에 비해 속도가 약간 느려지게 된다. JEUS Web Server 에선 내부적으로 효율적인 방법을 이용하지만, 한계가 있는 관계로 약간의 속도 저하는 발생할 수 있다.

두 번째로 Capacity 에 대한 내용으로 Log 를 남기는 공간에 대한 것이다. 접속자가 적은 곳에서는 Log File 이 얼마 남지 않겠지만 사용자가 아주 많은 곳에서는 하루에 Log 가 몇 백 Mbytes 에서 심한 경우 수 Gbyte 에 이르는 Log 를 남기게 된다. 이는 System 에 적당한 공간이 남아있지 않다면 불가능하기 때문에 이에 대한 관리가 중요하다.

그러나, 이들 정보들은 Web Server 를 관리하는 사람에겐 아주 중요하다. 자신이 운영하는 Web Server 에 들어온 사람들의 정보를 바탕으로 좀 더 효율적인 Web Service 를 구현할 수도 있고 Shopping Mall 등을 운영하는 사람들에겐 고객들의 정보를 알 수 있는 아주 귀중한 자료이다. 따라서 JEUS Web Server 를 이용하는 운영자들은 거의 Log File 을 특정 위치에 저장할 것이다.

그러면, 이제 JEUS Web Server 에서 남길 수 있는 Log File 에 대해서 알아보자. JEUS Web Server 에서는 기본적으로 두 가지 Log File 을 남길 수 있다. 사용자의 접속 및 관련 자료 접속 기록을 위한 Access Log File 과 System 에 혹시 발생할 수 있는 error 를 기록하는 Error Log File 이 있으며 각 Log File 을 기록하는데 있어서 우선순위가 존재한다. 이에 대한 설명은 다음과 같다.

4.7.1 Access Log File

Access Log File 은 Transfer Log File 이라고도 한다. 일반적인 사항을 모두 기록하며, 접속자가 들어와서 Web Server 에서 한 행동을 그대로 보여줄 수 있다. 그러므로 이 정보는 차후에도 많은 이용가치가 있기 때문에 아주 중요하다. JEUS Web Server 의 NODE, SVRGROUP, VHOST 등에 모두 설정이 가능하며 각 절에 모두 설정했을 경우 접속자의 접속서비스를 요청에 따라 그 우선순위가 SVRGROUP, VHOST, NODE 순서로 존재하며 가장 우선순위가 높은 절에 해당하는 Log File 에 그 내용이 기록된다.

4.7.2 Error Log File

Error Log File 은 Web Server 에서 발생하는 모든 에러와 접속실패에 대하여 에러가 발생한 시간과 에러의 내용을 기록한다. 이는 System 에서 발생할 수 있는 error 에 대한 기록이기 때문에 Web Server 에 문제가 생긴 경우 문제 해결에 큰 도움이 될 수 있는 정보를 제공하게 된다. 따라서, Critical 한 업무를 수행하는 경우에는 이를 저장하여 System 에 문제가 생길 경우 이를 통하여 문제 해결을 쉽게 할 수 있다. Error Log File 은 NODE 에서 설정 가능하다.

4.7.3 JEUS Web Server Log File Format

현재 JEUS Web Server 는 Common Log File Format 외에도 사용자가 직접 원하는 설정을 하여 기록을 할 수 있도록 하였다. 보통 운영자들이 관리하는 경우에 대부분 CLF(Common Log File Format)을 이용한다. CLF(Common Log File Format)는 Web Server 의 원조라 할 수 있는 NCSA 계열의 Web Server 에

서 사용하는 파일형식으로 현재 대부분의 Web Server 가 지원하고 있다. 물론 Web Server 마다 자체적으로 Log File 의 형식을 지원하고 있지만, 대부분의 Web Server S/W 의 제작사는 이 CLF 라는 표준 Log File 형식을 지원하고 있다. 현재 Apache, Netscape Enterprise, IIS 등 대부분의 Web Server 들이 이것을 기본 항목으로 하여 이용하고 있다. JEUS Web Server 에서는 이를 DEFAULT 라는 항목으로 설정하여 기본적인 것으로 지정하여 쓰고 있다. 이에 대한 설명은 다음과 같다.

W3C 에서 규정하는 CLF 의 형식은 다음과 같다.

4.7.4 Common Log File 포맷

Common Log File 포맷은 다음과 같다.

```
remotehost rfc931 authuser [date] "request" status bytes
```

각 항목은 다음과 같다.

`remotehost`

리모트 호스트 명(만약 DNS 호스명을 사용할 수 없거나 DNSLookup 이 off 이면 IP 주소)

참고: http://www.w3.org/Daemon/User/Config/General.html- DNSLookup

`rfc931`

사용자의 리모트 로그 명

`authuser`

인증된 사용자의 사용자명

`[date]`

요청한 날짜와 시간

`"request"`

클라이언트가 요청한 내용

`status`

클라이언트로 응답한 HTTP 상태 코드(참고:http://www.w3.org/Protocols)

`bytes`

전송한 문서의 컨텐츠 길이

위와 같은 각각의 정보를 저장하여 JEUS Web Server 환경 파일에 지정된 Log File 로 위들의 정보를 기록하게 되는 것이다. 이를 JEUS Web Server 에서 저장하여 기록한 예는 다음과 같은 형태의 Log 이다.

```
143.248.148.42 - - [13/Feb/2001:16:46:13 +0900] "GET / HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:14 +0900] "GET /index_pb.gif HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:18 +0900] "GET /index.html HTTP/1.1" 200
7118
143.248.148.42 - - [13/Feb/2001:16:46:18 +0900] "GET /usage.png HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:37 +0900] "GET /usage_200102.html HTTP/1.1"
404 148
143.248.148.42 - - [13/Feb/2001:16:47:21 +0900] "GET /index.html HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:21 +0900] "GET /usage.png HTTP/1.1" 200 2509
143.248.148.42 - - [13/Feb/2001:16:47:24 +0900] "GET /usage_200102.html HTTP/1.1"
404 148
143.248.148.42 - - [13/Feb/2001:16:47:55 +0900] "GET /index.html HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:55 +0900] "GET /usage.png HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:57 +0900] "GET /usage_200102.html HTTP/1.1"
200 30798
```

위의 Log 내용에서 맨 앞의 것은 접속한 사용자의 IP Address 를 나타낸 것이고, 뒤의 hyphen 들은 각각 rfc931 과 AuthUser 를 기록하여야 할 곳이다. 그러나, 사용자가 이에 관련된 정보가 없기 때문에 이를 hyphen 으로 기록한 것이다. 그리고 뒤에 있는 [ ] 안에 있는 정보가 사용자가 접속한 시간을 나타낸 것이고, 뒤이어 등장하는 " "안의 정보는 사용자가 요구한 Request URL 이고 뒤의 숫자는 그에 해당하는 Response Status Code 이다. 그리고 마지막으로 나타나는 숫자는 사용자에게 Server 가 전달한 Byte 수 이다.

JEUS Web Server 의 환경 파일에서 위와 같은 Log 정보를 얻기 위한 설정은 아래와 같다:

<<*http.m*>>

```
*NODE
webmain WEBTOBDIR="/usr/local/webserver",
      SHMKEY = 69000,
      DOCROOT="/usr/local/webserver/docs",
      PORT = "5469",
      LOGGING = "log1",
      ERRORLOG = "log2"
```

```
*LOGGING
log1  Format = "DEFAULT",
      FileName = "/usr/local/webserver/log/access.log",
      Option = "sync"
log2  Format = "ERROR",
      FileName = "/usr/local/webserver/log/error.log",
      Option = "sync"
```

우선, NODE 절을 보면 맨 하단에 Logging 라는 항목이 존재함을 볼 수 있다. 이것이 바로 Access Log 를 기록하기 위한 것이다. 이에 대한 설정은 밑의 Logging 절에서 log1 이라는 것으로 설정이 되어 있음을 볼 수 있다.

그리고 Error Log 는 ErrorLog 라는 항목으로 설정이 된다. 위의 예에서 NODE 절의 LOGGING 밑에 ErrorLog 라는 항목으로 설정이 되어 있음을 볼 수 있다. 이에 대한 설정은 역시 LOGGING 절에 log2 라는 것으로 설정이 되어 있음을 볼 수 있다.

JEUS Web Server 에서 Log Format 을 정하는 곳은 LOGGING 절이다. 이 곳에서 FORMAT 이라는 항목으로 이를 정할 수 있다. 위의 예에서 보면, Access Log 는 DEFAULT 로 설정되어 있고, Error Log 는 ERROR 라는 항목으로 설정되어 있다. 이 중 Access Log 의 DEFAULT 는 위에서 말한 CLF Format 을 그대로 준수하고 있다. 또한, ERROR 항목은 JEUS Web Server 에서 Error 를 보기 편하게 정리한 Format 을 가지고 있다.

그리고, 위와 같은 정해진 형식 이외에도 사용자가 원하는 형태의 정보를 얻기 쉽도록 특정 Field 를 추가하였다. 이는 마치 C Language 의 printf() 함수를 연상하면 쉽게 이해할 수 있다. printf 함수는 내부에서 %d %l %s %c 등등으로 설정된 값으로 변수의 값을 출력하게 된다. JEUS Web Server 에서도 이와 유사한 기능을 제공한다. 단, 내부적으로 결정된 출력 값은 C Language 에서 제공하는 것과 다르기 때문에 주의가 필요하다.

이에 대한 것은 아래와 같다.

%a : remote 의 IP 주소

%u : 인증이 된 요청을 하는 사용자

%t : 요청의 날짜와 시간

%T : 서버가 요청을 처리하는데 걸린 시간(초)

%r : HTTP 메소드를 포함한 요청의 첫번째 줄

%U : 요청한 URL

%s : HTTP 상태 코드

%b : 전송된 파일의 크기

%{Header}i : 클라이언트 요청 HTTP 헤더 예) %{Cookie}I

%v : HostName 필드에서 정의된 서버명

%p : 클라이언트 요청을 받는 TCP 포트 번호

이들을 이용하여 다음과 같은 형태의 Format 을 만들어 낼 수 있다.

```
log2  Format = "%h %l %u %t \"%r\" %>s %b",
      FileName = "JEUS Web Server/log/test.log"
```

4.8 JEUS Web Server Configuration 컴파일

본 장에서는 JEUS Web Server 환경파일을 작성한 후, 이를 실제 JEUS Web Server 에 전달 할 수 있는(Binary) 형태로 만들기 위한 Compile 작업에 대하여 설명할 것이다. 상당히 간단한 작업이지만, JEUS Web Server 를 구동하기 위해서는 반드시 필요한 작업이기 때문에 이를 반드시 숙지하기 바란다.

지금까지 JEUS Web Server 환경파일 작성법을 알아보았다. 작성된 JEUS Web Server 환경파일은 컴파일 작업이 필요하다.

왜 환경파일을 컴파일 해야 하는가? 보통 하나의 프로그램을 만드는 과정을 보면, 먼저 소스 프로그램을 작성하고 컴파일 하여 에러들을 수정한 후에 올바른 실행 파일을 만들게 된다. JEUS Web Server 환경파일도 실제 프로그램 작성 방법과 동일하다. JEUS Web Server 환경파일을 컴파일 함으로써 에러 없는 올바른 JEUS Web Server 환경파일을 만들 수 있게 된다.

JEUS Web Server 환경파일을 작성하면서, 항목에 적절하지 않은 값을 설정하는 등의 실수를 범할 수 있다. 그런데, 만약 컴파일 작업을 거치지 않고 이 환경파일로 JEUS Web Server 시스템을 기동 시켰다고 생각해 보자. JEUS Web Server 시스템은 기동 될 때, JEUS Web Server 환경파일을 토대로 환경을 설정하기 때문에, 에러가 존재하는 환경 파일이라면 기동이 되지 않거나 환경설정이 잘못되어 JEUS Web Server 시스템 동작 시 예상치 못한 오류가 생길 수 있다. 따라서 JEUS Web Server 환경파일의 컴파일 작업을 통해 JEUS Web Server 가 정상적으로 동작할 수 있도록 올바른 JEUS Web Server

환경파일을 만들어 주어야 한다. 즉, 컴파일 작업이란 사용자가 이해할 수 있는 text 형식으로 된 문서를 정해진 문법에 맞추어 작성하고, 이를 기계가 인식할 수 있는 이진 파일로 변환하는 것이며, 이 과정 중에 문법적 오류나 시스템 설정 오류를 검증할 수 있는 것이다.

컴파일 작업은 wscfl 명령에 의해 이루어 진다.

```
Wscfl -i config source file name [-o config binary file name]
```

Options:

`-i`

컴파일 대상이 되는 환경파일 이름을 지정한다.

`-o`

컴파일을 통해 만들어질 이진(Binary) 환경파일 이름을 지정한다. 지정하지 않으면 default 로 'wsconfig'라는 이름의 파일이 만들어진다.

컴파일시 에러가 발생하는 경우, 에러 메시지가 출력되며 성공적으로 컴파일 되면 ASCII 환경파일이 Binary 로 변환되어 JEUS Web Server 환경파일(Binary File)이 만들어진다.

여러 Node 로 JEUS Web Server 시스템이 구축이 되어 있는 경우 특정 Node 에서 중앙 관리를 하고자 하는 경우 각 Node 에 wsracd(Web Server Remote Access Control Daemon)가 미리 기동 되어 있어야 한다. wsracd 는 환경파일 컴파일, JEUS Web Server 시스템 부팅, JEUS Web Server 시스템 다운, 동적 환경 변경 등을 특정 Node 에서 한번의 명령어로 가능하게 한다. 특정 Node 에서 환경파일 컴파일 작업을 수행하게 되면 모든 Node 에 전달되며 모든 Node 에 이진 JEUS Web Server 환경파일이 전달된다.

이진 JEUS Web Server 환경파일은 JEUS Web Server 시스템 기동과 종료, Service 테이블 생성등에 참조되며 하나의 환경파일로 모든 Node 가 관리된다. 따라서 환경파일의 변경 작업에는 상당한 주의가 요구된다.

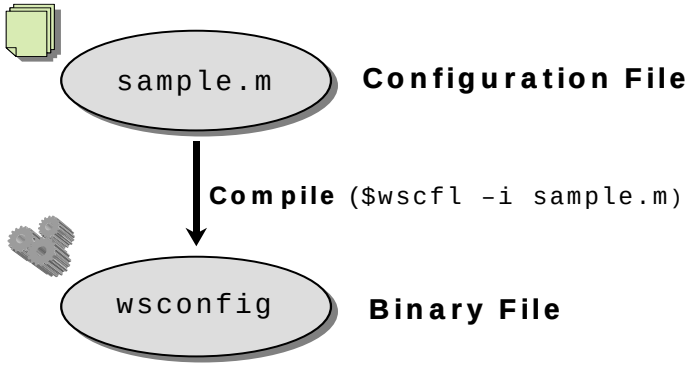

그림5. 환경 파일 컴파일 도식도

4.9 JEUS Web Server (WS engine) XML 환경 설정 파일 : (WSMain.xml)

4.9.1 소개

JEUS 5.0 이전 버전에서는 JEUS Web Server 의 환경 설정을 위해서 환경파일을 작성하였다. 위에서도 설명하였듯이 환경파일은 실제 JEUS Web Server 기동에 필요한 모든 정보를 포함하고 있는 파일로, 사용자가 이 파일에 작성한 내용을 Web Server 에 전달한다.

JEUS 5.0 부터는 Web Server 설정도 다른 engine(Servlet/JSP engine, EJB engine, JMS engine)들과 마찬가지로 XML 파일을 통해 환경을 설정하게되었다. 사용자가 WSMain.xml 을 통해 환경을 설정하면, JEUS 부트 시 이 파일을 읽어들여 내부적으로 configuration 파일을 작성하고 이를 컴파일하여 binary 파일을 만들게 된다.

따라서, JEUS 시스템의 Web Server 를 추가하고 설정하기 위해서는 두 개의 파일에 설정 데이터를 입력해야 한다.

- **JEUSMain.xml 파일**: JEUS 의 주 설정 파일로서 JEUS_HOME\config \<nodename> 디렉토리 아래에 위치하고 있다. 이 파일에는 몇 가지의 태그만 추가해주면 된다.

- **WSMain.xml 파일**: JEUS_HOME\config\<nodename>\<nodename>_ws _<enginename> 디렉토리 아래에 위치하고 있다. 이 파일은 Web

Server 의 실제 정보를 모두 담고 있지만 JEUSMain.xml 파일에서는 단순히 Web Server 의 존재여부만을 선언하고 있다.

4.9.2 JEUSMain.xml 에 Web Server (WS engine) 추가

JEUSMain.xml 에 Web Server 를 추가하는 방법은 JEUS Server 안내서에 잘 설명되어 있다. 여기서는 Web Server 를 추가하는 간단한 예를 보여준다.

<<*JEUSMain.xml*>>

```
<jeus-system>
    . . .
    <node>
                . . .
        <engine-container>
            . . .
            <engine-command>
                <type>ws</type>
                <name>engine1</name>
                <startup-mode>xml</startup-mode>
                <deploy-mode>xml</deploy-mode>
            </engine-command>
                  . . .
        </engine-container>
         . . .
    </node>
    . . .
</jeus-system>
```

위의 예는 “engine1”이라는 Web Server (WS engine)를 추가하고 있다. 다른 태그에 대한 정보와 JManager 툴을 사용하여 engine 을 설정하는 방법에 대해서는 JEUS Server 안내서를 참조하라.

그리고, 필수 XML 헤더가 위에 생략되어 있음을 알린다.

위의 <engine-command>태그를 JEUS 부트 시 정상적으로 작동하게 하려면, JEUS_HOME\config\<nodename>\<nodename>_ws_<enginename> 디렉토리가 반드시 존재해야 한다. 여기서 “nodename”은 JEUS 의 노드 이름이고, 이는 로컬 머신의 호스트 이름을 의미한다. “enginename”은 JEUSMain.xml 의 <engine-command> 태그 중 <name> 태그의 값을 가지고 있어야 한다. 위의 예에서 호스트 이름이 “enrique”이라고 가정하면, engine 이름은 “enrique_ws_engine1”이 된다.

JEUS_HOME\config\<nodename>\<nodename>_ws_<enginename>의 아래 에는 반드시 WSMain.xml 이 존재해야 한다. 이 파일은 다음 절에서 설명한다.

4.9.3 WSMain.xml 의 설정

WSMain.xml 은 Web Server(WS engine)의 모든 운영 설정들을 포함하고 있으며, engine 의 홈 디렉토리에 들어 있다("engine 디렉토리"라고 일컫는다). 앞에서 설명 했듯이 이 디렉토리는 다음과 같은 경로를 가져야 한다.

JEUS_HOME\config\<nodename>\<nodename>_ws_<enginename>

경로 포맷에 대한 정보에 대해서는 앞 절에서 설명하였다.

WSMain.xml 파일에는 <ws-engine> 태그 아래에 다음과 같은 Web Server 설정들이 포함되어 있다.

- **Domain** : 이 항목에서는 독립적인 JEUS Web Server 시스템 전체의 전반적인 환경에 대해 정의한다. Domain 의 이름은 호스트 이름과 함께 암호화 되어 JEUS Web Server 의 License 확인에 사용된다. Domain 은 아래에서 설명하게 될 Node, VHost, SvrGroup, Server, Service, Directory, URI, Alias, DirIndex, Logging, Access, Authent, EXT, SSL, ErrorDocument, Expires, TCPGW 을 하위 항목으로 갖는다.

- **Node** : Web Server 를 이루는 각 Node 들에 대한 구체적인 환경을 설정한다.

- **VHost** : Web Server 로 Virtual Hosting 이 필요한 경우, 이에 대한 환경 설정을 한다. Virtual Host 기능은 실제로는 하나의 Web Server 가 동작하지만 각기 다른 URL 로 다른 문서를 제공하도록 함으로써 마치 여러 개의 Server 가 Service 를 제공하는 것처럼 보이도록 하는 기능이다.

- **SvrGroup** : Web Server 를 통해 응용 Server Process 를 접근하는 경우 Server Process 의 논리적인 연관성에 따라 이들을 그룹으로 관리할 필요가 있게 된다. 이 항목에서는 이러한 그룹에 대한 환경 설정이 이루어 진다.

- **Server** : 이 항목에서는 실질적으로 제공하는 Service 들을 등록한다. Web Server 는 등록된 Service 만을 처리하기 때문에 새로운 Server 프로그램이 추가되는 경우는 반드시 등록하여야 한다.

- **Service** : 이 항목은 Web Server 를 통해 비즈니스 로직을 바로 수행할 경우에만 설정이 필요하다.

- **Directory** : Node 내의 특정 Directory 의 속성을 정하기 위한 환경설정을 한다.

- **URI** : Client 요구의 URI(Uniform Resource Identifier)값에 따라 이를 처리하는 Service 를 구분할 수 있도록 한다. 즉, 특정 URI 가 입력으로 들어온 경우, 이를 특정 Service 에서 처리하도록 할 수 있다는 것이다.

- **Alias** : 실제 Server 안의 물리적 Directory 경로와 URI 를 Alias 시키도록 설정할 수 있다. 즉, 어떤 특정한 URI 에 대한 요구가 들어오면 이를 실제의 물리적인 Directory 에 Mapping 시켜서 이곳에서 원하는 Resource 를 찾아 처리하게 하는 방식이다. 이는 사용자가 Document Root 에 상관없이 지정할 수 있기 때문에 관리적 입장에서 매우 편리할 것이다. 그러나, 이곳에 지정되는 물리적인 Directory 는 반드시 절대경로 이어야 한다.

- **DirIndex** : Client 가 요구한 index.html 과 같은 특정 파일이 존재하지 않는 경우, 전체 Directory 구조를 보여줄 수 있도록 설정한다.

- **Logging** : Client 의 요구 내역을 기록하는 형식을 지정한다. 접근 내역과 에러 내역이 따로 저장되며 저장 형식을 지정할 수 있다.

- **Access** : 클라이언트의 IP 에 따라 서버의 resource 에 대한 접근 권한을 제한하거나 허가하는데 사용된다. Access 절은 Direcory, URI, EXT 절에 적용될 수 있으며, 각각에서 정의한 resource 를 허용/제한하게 된다.

- **Authent** : Client 의 접근을 제한하기 위한 인증과정을 유저와 그룹 단위로 통제할 수 있도록 설정한다.

- **EXT** : Client 가 요구한 파일의 확장자명에 따라 처리 담당 Process 를 지정할 수 있다.

- **SSL** : Web Server 에서 사용할 SSL(Secure Socket Layer)의 기능을 설정하는 곳이다. 이곳에서 정의된 형태로 SSL 서비스를 하게 된다.

- **ErrorDocument** : Web Server 에서 에러 문제가 발생했을 때 다음과 같은 네 가지 방법으로 대응할 수 있다.

  1. 소스코드에 정의된 에러 메시지를 출력한다.
  2. 사용자가 정의한 에러 메시지를 출력한다.
  3. 로컬 URL 로 재전송한다.

4. 외부 URL 로 재전송한다

- **Expires** : 클라이언트 요청에 따라 전송되는 서버응답 헤더의 정보를 설정한다. 특정 Mimetype 문서 전송시 서버응답 헤더내에 전송되는 문서의 expiry date 를 설정한다.

- **TCPGW** : Client Request 를 listener 와 request 를 처리할 server 와 관련된 정보를 설정한다.

여기서는 WSMain.xml 의 상위 레벨 element 에 대해서만 간략하게 설명하였다. 각 element 들의 하위 element 에 대한 설명 및 WSMain.xml 파일의 상세한 설정 방법은 부록 E 와 부록 F 를 참조하기 바란다.

주의 : Web Server 는 한 node 에 하나만 설정할 수 있다.

5 JEUS Web Server 튜닝

5.1 소개

JEUS Web Server 는 상업용 Web Server 로, 대규모의 전문적인 웹 서비스의 운영을 위해 개발되었다. 전문가를 위한 서버인 만큼 사용자에 대한 자유도가 높고 각자의 환경에 맞는 최적의 미세조정을 제공한다. 바꾸어 말하면, Web Server 운영자가 각자의 환경에 맞게 서버를 Tuning 하여 사용하게 되면 엄청난 효율과 성능을 얻을 수 있다는 것이다. JEUS Web Server 의 이러한 특징을 적절히 활용하여 쓸 수 있도록 본 장에서는 Tuning 에 있어 가장 중요한 항목들과 그 적용사례를 살펴보도록 하자.

5.2 HTH 설정

앞에서 설명한 바와 같이 JEUS Web Server 는 특정 Process 에서 일반 Browser 에 관련된 연결을 모두 관리하고 있다. 이러한 구조는 하나의 Process 에서 모든 처리를 도맡아 하여 빠르게 작업을 할 수 있는 장점이 있으나 Process 의 안정성에 의해 시스템의 성능이 크게 좌우되므로 주의하여야 할 필요가 있다. JEUS Web Server 의 경우 역시 이러한 구조를 따르고 있으며 앞에서 등장한 HTH 란 Process 가 바로 이런 역할을 한다. 그러나, 많은 적용 사례와 수많은 Rush Test 를 거치고 개량하여 Process 의 안정성 면에서는 걱정할 문제가 전혀 없다고 해도 과언이 아니다.

단, 이 Process 의 관리가 성능에 영향을 줄 수 있으므로 이를 적당히 조절할 필요가 있다. 그럼 JEUS Web Server 에서는 어떤 식으로 이를 설정해 주어야 할까? 우선 대표적인 UNIX 의 운영 체제인 Solaris 7 의 예를 들어 설명하여 보겠다.

Solaris 7 은 현재 많은 SUN Server 사용자들이 이용하는 운영체제로 국내에서도 많은 사용자를 보유하고 있다. 또한 일반 UNIX 사용자들이 가장 많이 사용하는 대표적인 운영체제이기도 하다.

JEUS Web Server 에서는 HTH Process 를 통해서 사용자의 연결을 관리한다고 설명하였다. 즉, HTH Process 에 모든 사용자의 연결이 맺어지는 형식이

다. 이 때 한 가지 질문을 던질 수 있을 것이다. HTH Process 하나에 몇 개 정도의 연결을 맺는 것이 좋은가?

이와 같은 질문에 대한 답은 사람들이 이용하는 운영체제마다 다르게 될 것이다. 이는 각 운영체제 마다 Process 하나가 열 수 있는 연결의 수를 제한하고 있기 때문이다. 이제 우리가 가정한 Solaris 7 의 경우에 대한 답을 해 보자.

이 Solaris 7 이란 운영체제는 사용자에게 아주 많은 권한을 부여하였다. 과거 Solaris 6 이하의 Version 에서는 하나의 Process 가 최대 1024 개의 연결만을 열 수 있도록 운영체제에서 제한 하였다. 그러나, 현 Solaris 7 부터는 이에 대한 제약이 없어졌다. 이론상으로는 40 만개 이상의 동시 연결이 가능하도록 하였다. 그러나, 이는 어디까지나 이론적인 수치일 뿐, 실제 적용된 사이트에서는 이를 실제적인 수치로 제한할 필요가 있다. 과거 Solaris 6 의 경우 1024 개가 최대였기 때문에 동시 사용자가 1000 명이 넘는 경우 HTH Process 의 수를 2 개 정도로 늘려서 처리하였다.

또한, 하나의 Process 에서 너무 많은 처리를 하면 하나의 Process 가 많은 CPU 점유를 차지하게 되고, 또 Process 의 동작에 무리를 줄 수 있기 때문에 동시 사용자가 많은 경우 가급적 적당한 정도로 분리하여 실제 사이트에서 운영하여 왔었다. 보통 하나의 HTH Process 로 600 ~ 700 개 정도의 수를 처리하도록 하였다. 그러나 만약 Web Server 를 운영하고 많은 Service 를 추가로 이용하는 경우 HTH Process 의 수를 조금 더 늘리는 것이 좋다.

이러한 정보는 JEUS Web Server 를 Booting 하면, Booting 화면에 JEUS Web Server 가 자체적으로 운영체제에게 문의하여 자신이 받아 들일 수 있는 최대의 동시 사용자수를 출력하여 준다. 따라서 운영자는 동시사용자 수의 예측과, 이 정보를 바탕으로 HTH Process 의 적당한 수를 예측하여야 한다.

JEUS Web Server 설정파일에서 HTH Process 의 수를 설정하는 방법은 다음과 같다.

JEUS Web Server 설정 예제

```
*DOMAIN
JEUS_Web_Server
*NODE
webmain     WEBTOBDIR=”/usr/local/webserver”,
            SHMKEY=69000,
            DOCROOT=”/usr/local/webserver/docs”,
            APPDIR=”/usr/local/webserver/ap”,
            PORT=”5469,
```

```
HTH=2,
Method="GET,HEAD",
IndexName="index.html",
LOGGING="log1",
ERRORLOG="log2"
```

위와 같이 설정하면 JEUS Web Server 가 두개의 HTH Process 를 Booting 하게 된다. 따라서, 각각의 HTH Process 를 이용하여 사용자의 접속을 처리하게 된다. 각각의 HTH Process 는 각 Process 에 걸리는 Load 에 따라서 적당히 사용자의 요구를 분배하기 때문에 하나의 HTH Process 가 특별히 일을 많이 하거나 하는 등의 문제는 발생하지 않는다.

5.3 Server Process 설정

Apache 와 같은 Web Server 는 하나의 Process 에 HTML, CGI, SSI 등등의 모든 Service 처리 루틴이 포함되어 있다. 따라서 사용자가 접속하는 경우 하나의 Process 에서 모든 처리가 끝난다. 이는 각 사용자의 요구 순간 마다 Process 가 접속을 연결 받아 처리하여, 결국 사용자당 담당 Process 가 연결되는 구조에서는 필연적인 것이다. 이런 구조에서는 하나의 Process 에서 모든 사용자의 Request 를 처리할 수 있다는 장점은 있으나, 사용자가 쓰지도 않는 기능들이 의미 없이 설정되어 있을 수도 있다.

간단한 예를 든다면 다음과 같다.

5.3.1 예제 Sample

만약, 어떤 Shopping Mall 을 운영하는 사람이 자신의 Web Server 를 운영한다고 생각하여 보자. 이 경우 운영자는 자신이 운영하여야 할 Web Server 를 위하여 HTML 문서들을 만들고 이에 들어가는 멋진 그림들이나 동영상들, 그리고 간단한 Script 등을 포함할 수 있다. 그러나 이러한 정적인 내용물로는 사용자들을 만족하게 할 수 도 없고 사용자가 물건을 사거나 결제를 할 때 필요한 기능을 이용할 수가 없다. 따라서, CGI 나 Servlet 혹은 기타 다른 Application Program 등을 이용하여 이를 처리하여야 한다. 예전 같은 경우에는 대부분의 Program 들이 CGI 로 구성되어 있었다. 가장 간단하고도 기능은 좋았기에 많은 개발자들이 이를 선호하였다. 그러나 성능상의 문제가 많은 관계로 현재는 거의 자취를 감추고 말았다. 따라서 현재는 소규모의 사이트나, 예전에 개발된 Program 등을 이용하는 경우를 제외하고는 거의 보기 힘들다. 이러한 이유 등이 있지만 Shopping Mall 운영자는 간단하게 만들 수 있다는 장점과 초기에 사용자가 그리 많지 않을 것이란 예상을 하고 CGI 를 도입하여 Program 을 개발하였다고 하자. 그리고 혹시 있을 문제를 대비하여

성능상의 문제를 위하여 사람들이 많이 이용할 것 같은 Program 은 Servlet 등을 이용하여 개발하였다고 하자. 그리고 또 SSI 를 이용하여 사용자에게 멋진 HTML Page 를 보이고자 한다.

이러한 개발 계획을 세워 장기간의 시간을 두고 Application Program 을 개발하여 운영자는 드디어 Shopping Mall 을 개장하였다. 이 때 Web Server 로 Apache 를 이용한다면 아마도 운영자는 사용자수를 대략 예측하여 Process 의 수를 결정할 것이다. 만약 동시 사용자가 100 명 정도라면, Apache Process 의 수를 100 정도로 결정할 것이다. 이 경우, 각 100 개의 Process 에 HTML, CGI, SSI, Servlet 에 관계된 내용들이 모두 포함되어 있다. 결국 하나의 Process 에 모든 기능들이 포함되어 큰 Memory 양을 차지하고 있게 된다. 이 때 만약 사용자들이 정작 들어와서 이용하리라 예상했던 CGI 는 거의 쓰지 않고 대부분 HTML 만을 본다고 생각하자. (실제 대부분의 경우가 이렇다고 보면 된다.)

대략, 100%의 사용자를 기준으로:

- 60%가 HTML 이용자
- 20%가 Servlet 이용자
- 10%가 CGI 이용자
- 10%가 SSI 이용자라는 가정을 세우자.

위의 가정이 정확하지 않을 수도 있지만, 대부분의 경우 하나 혹은 두개 정도의 Service 에 사람이 몰리는 경우가 일반적이다.

다시 위의 가정으로 돌아가서, 운영자는 이미 100 개의 Process 를 설정하였다. 이는 사용자가 동시에 100 명 정도가 들어오는 경우 반드시 필요한 수치이다. 하지만 이 때 위와 같은 사용자 분포도를 보인다면, 100 개의 Process 에서 60 개는 거의 HTML 만을 Service 하고 20 개는 Servlet 을 Service 하고, 10 개가 CGI 를 Service 하고 10 개가 SSI 를 Service 를 하게 된다. 이 때 각각의 Process 에 포함된 다른 기능들은 의미 없이 Memory 만 차지하고 있다는 결론이 된다. 게다가 사용자가 HTML 만을 반복적으로 이용하는 경우, 이런 현상은 더더욱 두드러지게 나타난다. 이는 결국 Memory 를 효율적으로 운영할 수 없다.

이런 경우에 JEUS Web Server 는 위와 같은 문제를 어떻게 해결하는가?

JEUS Web Server 에서는 각각의 HTML, CGI, SSI 등의 Service 들이 별도의 독립적인 Process 로 설정된다. 결국 하나의 Process 에서 모든 처리를 하는

것이 아니라, 각각의 Service 를 처리하는 것을 분리하여 별도로 각각의 수치를 조정할 수 있게 하였다.

따라서 위와 같은 경우 운영자가 HTML 을 사용자가 많이 사용한다는 것을 알게 된다면, HTML 을 처리하는 Process 들을 많이 설정하면 된다. 그리고 다른 Process 들은 적게 설정하여 불필요한 Memory 의 낭비를 막을 수 있다.

Apache Web Server 는 HTML 을 처리하건, CGI 를 처리하건 Process 의 조정이 불가능하다. 하지만 JEUS Web Server 의 경우엔 특정 Service 가 분리되어 있기 때문에 이를 조정할 수 있다. 따라서 위의 경우에 JEUS Web Server 에선 HTML 에 관련된 Process 를 6 개 정도 띄우고 Servlet 에 관련된 것을 2 개 정도 그리고 CGI 와 SSI 를 각각 1 개 정도 띄워 준다면 System 에서 가장 최적인 환경에서 Service 를 할 수 있을 것이다.

이를 실제로 JEUS Web Server 설정파일에서 적용하는 것은 각각의 SERVER 절에서 MaxProc 와 MinProc 를 통하여 이루어진다. 이에 대한 예는 다음과 같다.

```
*SERVER
html  SVGNAME = htmlg, MinProc = 6, MaxProc = 10
jsv1  SVGNAME = jsvg, MinProc = 2, MaxProc = 10
cgi   SVGNAME = cgig, MinProc = 1, MaxProc = 10, SvrCPC = 4
ssi   SVGNAME = ssig, MinProc = 1, MaxProc = 2
```

위의 예에서 각 SERVER 절에 있는 MinProc 가 처음 JEUS Web Server 가 Booting 될 때 시작되는 Process 의 수이다. 따라서 위의 경우 HTML(html)은 6 개, Servlet(jsv1)은 2 개, CGI(cgi)는 1 개, SSI(ssi)는 1 개를 초기값으로 설정한 것이다. 뒤에 있는 MaxProc 는 최대로 이용할 수 있는 Process 의 수이다. 이는 동적으로 JEUS Web Server 를 조정하는 것이 가능하기 때문에 필요한 Field 이다. 즉, 사용자가 초기값으로 설정한 후에, 약간의 변화가 생겨 Process 의 조정이 필요한 경우 MinProc 와 MaxProc 의 범위 안에서 조정하는 것이 가능하다. 이는 wsadmin 이라는 관리자용 Tool 을 이용하면 된다. 이에 대한 설명은 JEUS Web Server 관리자용 Tool wsadmin 을 참조하면 된다.

6 JEUS Web Server 제어

6.1 소개

본 장에서는 JEUS Web Server 를 실제 기동하고 종료하는 작업에 대한 설명을 할 것이다.

6.2 JEUS Web Server 기동

JEUS 에서는 JEUS Web Server 를 JEUS 의 웹 서버 엔진으로 본다.

그리고 JEUS Web Server 를 기동하는 방법도 네 가지가 있는데, 이를 실행시키는 방법을 알아보자.

1. JEUS 와 JEUS Web Server 의 실행에 필요한 모든 환경파일을 설정하고 컴파일 한 후 JEUS 를 BOOT 시키면 JEUS Web Server 도 JEUS 엔진의 일부이기 때문에 JEUS 웹 서버라는 이름(node 명_ws_Engine 명)으로 BOOT 된다.

2. jeusadmin 콘솔 명령상에서 'jeusadmin node 명 starteng node 명 _ws_Engine 명' 명령으로 JEUS Web Server 엔진을 Start 시킬 수 있다.

3. JEUS Administration Tool 을 이용하여 JEUS Web Server 를 BOOT, DOWN 시킬 수 있다.

4. 마지막으로, 스크립트 명령어인 wsboot, wsdown 을 이용해서 JEUS Web Server 를 기동 및 종료 시킬 수 있다.

참조 : JEUS 부팅에 관련된 보다 자세한 설명을 원한다면, JEUS Server 안내서를 참조하기 바란다.

6.3 JEUS Web Server 제어

동적인 시스템 관리를 위하여 JEUS Web Server 시스템에서는 `wsadmin` 이라는 프로그램을 제공한다.

wsadmin 프로그램은 UNIX 환경의 shell 과 비슷한 Command Interpreter 이다. 즉, 항상 프롬프트 상태로 대기중이다가 입력되는 명령어를 해석하여 이를 실행하게 된다. Web Server 가 존재하는 경우에는 웹을 통해서도 관리가 가능하다

제공되는 Service 들 중 특정 Service 에 대하여 현재 처리 상태 즉, 몇 건의 Service 를 처리해 왔으며, 평균 처리 시간이 얼마나 되며, 몇 건의 요청이 대기하고 있는지, 그 Service 를 제공받기 위해서는 얼마나 기다려야 하는지 등의 정보를 확인할 필요가 있다

wsadmin 에서 사용 가능한 명령어들은 다음과 같다. [표 1]:

표 1. wsadmin 에서 사용 가능한 명령어.

명령어	내용
suspend	동작 중인 Server Process 중지.
resume	중지된 Server Process 재개
qp	큐에 적체된 요청 삭제.
set	현재 설정된 환경 값을 동적으로 변경
rbs	Server 프로그램 교체
ds	접속중인 웹 브라우저를 강제로 해제.
logstart	logging 시작
logend	logging 종료

명령어들에 대한 전반적인 정보를 얻고자 한다면, 부록 A 를 참조하기 바란다.

7 JEUS Web Server 관리

7.1 소개

JEUS Web Server 시스템이 동작하고 있다면 시스템의 관리가 필요하다. 예를 들어, JEUS Web Server 시스템의 종료 및 기동의 절차를 거치지 않고도 현재 환경 설정에 대한 정보를 확인해 보고 이를 동적으로 변경할 필요가 있다.

또한 제공되는 Service 들 중 특정 Service 에 대하여 현재 처리 상태 즉, 몇 건의 Service 를 처리해 왔으며, 평균 처리 시간이 얼마나 되며, 몇 건의 요청이 대기하고 있는지, 그 Service 를 제공받기 위해서는 얼마나 기다려야 하는지 등의 정보를 확인할 필요가 있다.

이 정보에 따라 그 Service 를 제공하는 Server Process 를 더 기동 시켜 준다든지 아니면 필요 이상의 Server Process 가 동작 중일 경우 몇 개는 종료 시키는 등의 조치도 필요하다. 이러한 동적인 시스템 관리를 위하여 JEUS Web Server 시스템에서는 `wsadmin` 이라는 프로그램을 제공한다.

7.2 wsadmin 을 이용한 관리

wsadmin 에서 관리에 사용 가능한 명령어들은 다음과 같다. [표 2]:

표 2. wsadmin 에서 관리를 위한 명령어.

명령어	내용
wi	JEUS Web Server 시스템 정보 확인
ci	접속 웹 브라우저 확인
si	Server 정보 확인
history	명령어 저장 기능

명령어	내용
!	직전 명령어 반복
config (cfg)	환경설정 내용 조회
stat (st)	Process 및 Service 상태 통계

명령어들에 대한 전반적인 정보를 얻고자 한다면, 부록 A 를 참조하기 바란다.

8 JEUS Web Server 보안 설정

8.1 소개

JEUS Web Server 는 기본적으로 Authentication 과 SSL 을 지원한다. 이들은 기존의 다른 Web Server 들에서도 기본적으로 지원하는 것으로 Web 상에서 Security 를 보장하기 위하여 가장 많이 쓰이는 방법들이다. 근본적으로 Open Network 을 지향하는 Internet 에서 사용자들의 정보를 보호하고, 자료의 유출을 막기 위한 방법들인 이들은 전자 상거래 등의 거대한 규모의 Site 에서 반드시 검증하고 이용하여야 할 방법들이다.

8.2 인증방법

Authentication(인증)의 방법은 매우 단순하다. 사용자가 사용자명과 패스워드를 Web Server 에 보내고 Web Server 는 사용자가 접근 권한을 가지고 있는지 확인하기 위하여 이름과 암호화된 패스워드가 있는 파일을 열고 위의 정보들을 검색한다. 그리고 검색된 정보에서 사용자명과 패스워드의 일치를 확인하고 문제가 없으면 사용자에게 정당한 권리를 부여하는 방식이다. 이는 개인별로 나누어서 접근 권한의 설정도 가능하고 몇몇 사람들을 그룹으로 묶어서 접속 권한을 주거나 거부하도록 설정하는 것도 가능하다. 물론 전체 모든 사용자들에 대한 설정도 가능하다. 실제 Browser 와 Server 간의 작동은 다음과 같다.

먼저 사용자가 어떤 Server 에 접속하고 이어서 어떤 자료를 요청한다고 하자. 이 때 사용자가 요청한 자료가 Server 가 보기에 중요한 자료이기 때문에 특정 사용자들만 보거나 실행할 수 있는 권한이 있다면, Server 는 사용자에게 Authentication Required [HTTP 401] 신호를 전달한다. 그러면 사용자는 Web Server 에 자신의 사용자명과 패스워드를 같이 보내고 이를 가지고 Web Server 는 사용자의 인증을 수행한다. 이 때, 사용자의 사용자명과 패스워드가 Internet 상으로 나가게 되는데 만약 이에 대한 정보를 누군가 중간에서 볼 수 있다면 문제가 될 수 있다. 이에, 사용자의 사용자명과 패스워드를 암호화하여 보내는 방법을 취하고 있다.

8.3 SSL (Secure Socket Layer)

JEUS Web Server 에서는 현재 많이 퍼져 나가고 있고, 앞으로 계속 늘어날 전자 상거래 Site 를 위해서 안전한 보안 방식인 Secure Socket Layer (SSL)을 제공한다. JEUS Web Server 에서 구현된 SSL 을 이용하기 이전에 먼저 SSL 에 대한 것을 짚고 넘어 가자. SSL 은 Web Server 가 제공하는 것이기 때문에 SSL 에 대한 내용만 이해한다면, 사용하는 방법은 아주 간단하다.

8.3.1 SSL v3.0

Netscape 사에서 개발한 암호화 전송 프로토콜 (RC2, RC4 로 암호화) X.509 Certificate 를 이용하여 서버와 Client 간의 인증을 한다. 40bit 키를 사용하는 SSL 은 보안에 있어서 신뢰를 주지 못하고, 128bit 키의 SSL 을 사용하여야 하며 현재 SSL version 은 3.0 이다.

SSL 은 Layer 를 기반으로 하는 Protocol 이다. 각 Layer 의 Message 는 length, description, content field 들을 포함할 수 있다. SSL 이 전송해야 할 Message 를 가질 때, SSL 은 그것을 제어 가능한 크기의 block 으로 나누고, 선택적으로 Data 압축과정을 거친 후, MAC 을 추가하고 암호화를 한 후, 결과를 전송한다. Data 가 도착하게 되면 복호화를 하고 MAC(Message Authentication Code)을 Verify 한 후, 압축된 Data 를 원 상태로 만들고, Block 을 재배열하여 얻어진 Message 를 상위레벨로 넘겨주게 된다.

SSL 을 사용하기 위해서는 우리가 흔히 사용하는 URL 표기 방식인 "http://*" 대신에 "https://*"을 사용해야 한다. (물론 SSL 을 지원하는 Web Server 에 연결을 할 경우에만 가능하다). 보통의 경우(http)에 Port 번호 80 을 사용하는데 SSL 을 사용할 경우(https)에는 Port 번호 443 을 호출하므로 하나의 브라우저를 이용하여 "http://*"와 "https://*"를 동시에 사용할 수가 있다.

SSL 이 작동하기 위하여 필요한 사전 작업들이 "handshake" 과정에서 수행된다. 이 작업들을 정리하면 다음과 같다. Client 와 서버가 서로 자신의 인증서(Certificate)를 교환한다. 그리고 각각은 자신이 받은 인증서에 기록되어 있는 유효기간, 서명을 확인한다. Client 는 이후에 수행될 암호화와 메시지 인증 코드(MAC : Message Authentication Code)의 생성에 필요한 난수(비밀키)를 생성하여 이것을 서버의 공개키로 암호화하여 서버에게 전송한다. 서비스를 받는 동안 사용할 암호화 알고리즘과 해쉬 함수의 종류를 결정한다. 이 과정에서는 Client 는 자신이 받아 들일 수 있는 암호화 알고리즘의 리스트를 서버에게 제시하고 이들 중에서 하나를 서버가 선택한다.

8.3.2 SSL vs. SHTTP

SSL 과 SHTTP 는 가끔 그 관계가 혼동되는데 사실 이 두 가지 프로토콜은 그 성격이 많이 다른 것이다. SHTTP 는 EIT(Enterprise Integration Technology)에서 개발한 프로토콜로서 기존의 HTTP 프로토콜에 보안기능을 갖게 하기 위하여 몇 가지의 헤더를 추가한 것이다. 앞서 설명한 것과 같이 SSL 은 서비스를 단위로 암호화와 인증을 받지만 SHTTP 의 경우에는 전달되는 메시지를 단위로 암호화와 인증을 받는다. SSL 과 비슷한 점이 있다면 Client 가 서버에게 SHTTP 를 이용한 커넥션을 만드는 과정에서 다양한 종류의 암호화 기법들 중에서 하나를 선택할 수 있고 암호화의 정도를 선택할 수 있다는 점이다. SHTTP 가 지원하는 암호화 기법의 특징을 정리하면 다음과 같다.

알고리즘:

- RSA, Diffie-Hellman 또는 Kerberos 인증기법을 사용한다.
- 공개키 인증서(certificate) 양식 : X.509 또는 PKCS-6 를 사용한다.
- 디지탈 서명 알고리즘 : RSA 또는 DSA(Digital Signature Algorithm)를 사용한다.

8.3.3 SSL Encryption

40bit 냐, 128bit 냐 하는 것은 주어진 Data 를 Encoding 하는데 필요한 Key 의 길이이고, 이것이 길수록 훨씬 안전하다. 보통 SSL 등을 지원하는 apache-ssl 이나 Netscape, IIS 등은 자체적으로 128bit 를 지원하나, 이러한 128 bit Encryption 을 지원하기 위해서는 Server 뿐만 아니라 Client 역시 이를 지원해야 한다. 그러나 미국에서 만들어진 대부분의 소프트웨어(IE5.0 이나 Netscpe4.5 등)들은 128bit 수출금지법에 의해 국내에서는 40bit 로만 동작하게 되어, 미국에서 사용하는 경우라면 MS Site 에서 128bit upgrade patch 를 받을 수 있으나, 국내에서는 그 사용이 불가능하다. (일부 자료실에 등록되어 있는 경우도 있다.) 따라서 국내 browser 에서 128bit strong encryption 을 지원하기 위해서는 Browser 와 독립적인 국내에서 제작된 128bit Gateway Software(내부적으로는 proxy 기법을 이용)를 이용하는 것이 보다 보편적인 추세이며 이 경우, 특정한 국내 site 접속에는 문제가 없으나 국제적인 Internet Banking 등을 이용하기 위해서는 여전히 한계가 있다.

8.3.4 Ciphers

Ciphers 는 암호화에 사용되는 알고리즘이고, 더욱 더 안전하고 강력한 것들을 선택할 수 있다. 일반적으로 Ciphers 가 암호화할 때 더 많은 비트를 사용할수록 그 데이터를 해독하기가 더 어렵고 어떠한 양방향 암호화 과정에서도 양자는 같은 Cipher 를 사용해야 한다. 이러한 Cipher 에는 여러 가지 종류

가 있어서 서버는 가장 많이 알려진 것을 사용할 수 있어야 하고, Client 가 서버와의 SSL 접속을 시작하면 정보를 암호화하는데 선호하는 Cipher 를 서버에 알리게 된다.

8.4 인증 서비스

8.4.1 인증기관(Certificate Authority)

인터넷과 같은 개방형 통신망에서 일어날 수 있는 사용자 데이터의 보안 및 사용자 신원인증을 위한 방법으로는 공개키 암호화 알고리즘이 있다. 이 알고리즘은 데이터를 암호화하는 키와 암호화된 문을 해독하는 키가 쌍으로 구성된다.

비밀키는 해독할 사람 본인이 가지고 있고, 공개키는 공개하여, 다른 사람들이 비밀키의 소유자에게 메시지를 전할 때 그 공개키로 암호화해서 보내는 방식이다. 이 방법에서 문제가 되는 것은 그 공개 열쇠가 정말로 사용자 본인의 것인가 아닌가 하는 점인데, 이를 확인하기 위해 공개 키를 증명해주는 기관을 따로 두게 되며, 이를 인증기관(Certification Authority, CA)이라 한다.

이와 같이 메시지 송신자와 수신자가 모두 신뢰할 수 있는 제 3 의 인증기관을 바탕으로 그 인증기관에서 발행하는 사용자 인증서를 상호 교환 및 확인을 통하여 원격지 상대방의 신원인증 및 송수신되는 데이터의 보안을 하게 된다. 인증기관에서 발행하는 인증서에는 인증기관 인증서(CA Root 인증서), Web Server 인증서, 사용자 인증서의 세 종류로 구분한다.

8.4.2 인증서(Certificate)

인증서란 어떤 소프트웨어나 어떤 기관, 혹은 개인을 보증하는 Q 마크와 같은 것이다. 잘 모르는 신제품에 대하여 고객은 KS 나 Q 마크를 믿고 그 제품을 구입하게 되는데 인터넷에서도 이와 같은 개념으로 도입된 것이 인증서인 것이다. 반대로 상점의 입장에서 보면 물건을 사러 온 고객의 옷차림이나 생김새로 믿어도 좋은 사람인지 무엇을 훔치러 들어온 사람인지를 대략적으로 알 수가 있고 고객의 신분증을 보고 신용카드 거래를 허용하는 일을 할 수도 있다. 이와 같이 인증서는 사용하는 입장에서 보면 '서버 인증서'와 'Client 인증서' 2 가지로 나누어 볼 수 있고 tree 구조를 따라 보다 상위기관에서 그 신뢰도를 인증하는 계층구조로 이루어져 있는 것이 보통이며 사용자 데이터 보호와 사용자 신원의 추가 확인용으로만 사용될 뿐, 전자 서명용으로 사용할 때는 전자서명법에 의한 법적효력을 갖지 않는 것이 보다 일반적이다.

이러한 온라인 인증서는 인증하고자 하는 대상에 관한 몇 가지 내용을 기술한 문서라고 볼 수 있는데 개인적으로 만들 수도 있고 발급기관에 요청할 수

있으며 그 형식은 ITU (International Telecommunication Union)에서 제정한 X.509 의 양식 또는 RSA Data Security 사에서 제정한 PKCS-6 의 양식 및 PEM 을 따르는 인증서를 사용한다. 이러한 인증서는 cert.crt 와 같은 형식으로 배포될 수 있으며 이것은 Private Key 를 제외하고 배포되는 Public Key 라고 생각할 수 있다. 다음은 PKCS-6 의 예이다.

```
Certificate:
Data:
Version: 0 (0x0)
Serial Number: 02:41:00:00:01 -------------------------- ①
Signature Algorithm: MD2 digest with RSA Encryption ----- ②
Issuer: C=US, O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority --------------- ③
Validity:
Not Before: Wed Nov 9 15:54:17 1994
Not After: Fri Dec 31 15:54:17 1999 -------------------- ④
Subject: C=US, O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority
Subject Public Key Info:
Public Key Algorithm: RSA Encryption
Public Key:
Modulus:
00:92:ce:7a:c1:ae:83:3e:5a:aa:89:83:57:ac:25:
01:76:0c:ad:ae:8e:2c:37:ce:eb:35:78:64:54:03:
e5:84:40:51:c9:bf:8f:08:e2:8a:82:08:d2:16:86:
37:55:e9:b1:21:02:ad:76:68:81:9a:05:a2:4b:c9:
4b:25:66:22:56:6c:88:07:8f:f7:81:59:6d:84:07:
65:70:13:71:76:3e:9b:77:4c:e3:50:89:56:98:48:
b9:1d:a7:29:1a:13:2e:4a:11:59:9c:1e:15:d5:49: dd:2d:d6:c8:1e:7b
Exponent: 65537 (0x10001)
Signature Algorithm: MD2 digest with RSA Encryption
Signature:
88:d1:d1:79:21:ce:e2:8b:e8:f8:c1:7d:34:53:3f:61:83:d:
b6:0b:38:17:b6:e8:be:21:8d:8f:00:b8:8b:53:7e:44:67:1:
22:bd:97:27:e0:9c:85:cc:4a:f6:85:3b:b2:e2:be:92:d3:e:
0d:e9:af:5c:0e:0c:46:95:ff:a1:1c:5e:3e:e8:36:58:7a:7:
a6:0a:f8:22:11:6b:c3:09:38:7e:26:bb:73:ef:00:bd:02:a:
f3:14:0d:30:3f:61:70:7b:20:fe:32:a3:9f:b3:f4:67:52:d:
b4:ee:84:8c:96:36:20:de:81:08:83:71:21:8a:0f:9e:a9
```

위의 인증서에서,

①에서는 인증서의 시리얼 번호를 표시하였고.

②에서는 인증서에서 RSA 암호화 알고리즘과 MD2(Message Digest 2)라는 메시지 압축 알고리즘을 사용하였음을 나타내고.

③에서는 인증서를 발급한 기관에 관한 정보를 나타내었다. 여기에서 C = country, O = Organization, OU =Organization Unit 를 의미한다.

④에서는 이 인증서의 유효기간이 1994 년 11 월 9 일 15 시 54 분 17 초에서부터 1999 년 12 월 31 일 15 시 54 분 17 초까지라고 표시하고 있다. CA(Certificate Authority)는 인증서를 발급해주는 기관의 서버를 의미한다.

8.4.3 인증서 발급 정책(Certificate Issuing Policy)

위에서 살펴본 것과 같이 보다 확실한 보안통신을 위해서는 서버뿐만 아니라 Client 입장에서의 인증서가 필요하다. 보다 자세히 설명하자면, Client 의 경우에 "정말로 내가 요청한 서비스를 해줄 적법한 서버인가 ?"를 확인 하는 것이 필요할 것이고, 서버의 경우에 "정말로 서비스를 요청한 실체가 적법한 Client 인가?"를 확인하는 작업이다. 보통 서버는 의무적으로 인증서를 제공을 하고, Client 에서 인증서를 요구할 것인가 하는 것은 발급하는 정책에 따라 여러 가지가 있을 수 있는데, 일반적으로 다음의 4 가지 형태로 존재하게 된다:

0 인증과정 필요 없음. No certificate required.

1 Client 는 올바른 인증서를 제공할 수도 있다. 인증서가 제공되면, 서버가 가지고 있는 인증서와 일치하는 인증기관(Certification Authority)에서 인증한 것이어야 한다.

2 Client 는 올바른 인증서를 반드시 제공해야 한다.

3 Client 는 올바른 인증서를 제공할 수도 있다. 그러나 반드시 서버의 인증기관과 일치할 필요는 없다.

일반적으로, 영화표 예매나 사이버 증권사와 같은 곳에서는 고객의 ID 와 Password 만 있으면 서버의 인증서를 바탕으로 보안 통신이 이루어지게 되는 type 0 형식이고, 한국통신에서 진행하는 banktown 21c 프로젝트에서는 보다 확실한 보안을 위한 type1 의 형식이다. 서비스를 이용하기 위해서는 웹사이트에서 "인증서 제출"이란 버튼을 눌러 인증서를 제출하면(일반적으로는 Browser 에서 이것을 제공하게 할 수도 있다.) 전자통장이 실행되고, 동시

서버에서 인증서를 발급 받아 상호간의 인증(Mutual Authentication)을 획득하게 되는 것이다. type0 과 같이 단순히 패스워드를 입력하는 방법보다는, 인증서를 이용하는 방법이 인증서와 패스워드를 동시에 이용하기 때문에 더욱더 안전하다고 할 수 있다. 우리는 흔히 은행의 ATM 단말기를 이용할 때 은행에서 발급한 카드를 넣고 패스워드를 입력한 후에 현금을 인출한다. 이러한 과정에서 카드가 인증서에 해당한다고 볼 수 있다. 따라서 우리는 이 카드를 입력함으로써 은행의 서버에게 우리 자신의 실체를 알리고 카드를 입력한 사람이 적법한 사용자임을 패스워드 입력을 통해서 밝히는 것이다. 이것은 디지탈 서명에서 사용한 방법과 똑같은 것이다.

8.4.4 Verisign 에서 서버 인증서 발급 받기

Verisign 은 세계적으로 가장 유명한 사설인증 기관이다.Verisign homepage 인 www.verisign.com 의 SSL Server Certificate 를 선택하면 인증서를 받을 수 있다. 인증서는 돈을 주고 영구적으로 권한을 살 수도 있고, 일정기간(14 일) trial 로 사용해 볼 수도 있는데, 실제로 구입하기 위해서는 약간 더 복잡한 과정이 필요하다. 인증서란 한 마디로 말하여 어떤 대상을 인증기관에서 신용을 보증하는 것이기 때문에 verisign 사는 인증서를 요청한 기관이나 업체에 사업허가서 등의 제반 서류를 요청하게 되는데, D&B 라는 Business Database 에 등록되어 있다면 이러한 과정이 훨씬 간단해 질 수 있고, 현재는 주로 미국내의 기업들만이 등록되어 있다.

위에서 설명한 것 외에도 SSL 은 더 복잡한 구조와 인증 방법을 가지고 있다. 또한, SSL(Secure Socket Layer)은 TCP Protocol 과 HTTP Protocol Level 사이에 있는 Data 를 암호화할 수 있는 Layer 이기 때문에, SSL 의 암호화를 통하여 Data 가 Internet 등의 누구에게나 Open 된 Network 으로 나간다 할지라도 암호를 해독할 가능성이 거의 없기 때문에 Data 에 대한 보안성이 생기게 된다. 따라서 이는 다시 한 번 반복하지만, 보안성이 특히 중요한 전자상거래 등에서 많이 쓰이게 되며, 국내에서도 대부분의 전자상거래에서 많은 사용자들이 이용하고 있다.

하지만, 이는 TCP Protocol 과 HTTP Protocol Level 사이에서 Data 를 암호화해야 하기 때문에, 성능상에 문제점을 초래할 수 있다. 특히 SSL 을 지원하는 Apache 등 타 Web Server 에서는 SSL Package 와 복잡한 연동 문제로 인하여 처리 속도가 느려지는 문제점이 발생하게 된다. 왜냐하면 이들 Web Server 가 SSL 을 외부의 함수로 이용하여 자신들의 내부 Engine 과 결합성이 떨어지기 때문이다. 이에 JEUS Web Server 에서는 이들을 내부 Engine 과 결합하여, 성능에 문제가 될 수 있는 모든 요인들을 제거하고 효율성에 초점을 맞추었다.

실제 SSL 의 이용은 매우 단순하다. 현재 사용자들이 많이 이용하는 Browser 가 거의 모두 SSL 을 지원하기 때문에 https 로 시작하는 사용자 요구들을 스스로 SSL 로 간주하여 Web Server 로 보내게 된다. 또한, Web Server 에서는 이런 요구를 받아 위에서 설명한 인증 방법 등을 동원하여 사용자를 인증하게 된다. 이는 모두 Web Server 와 사용자 Browser 내에서 이루어지는 일로 사용자가 복잡하게 직접 관여할 필요는 없다.

8.5 Authentication 과 SSL 의 이용

8.5.1 Authentication 의 설정

JEUS Web Server 에서 Authentication 을 설정하는 방법은 앞의 Configuring JEUS Web Server 절에서 설명하였다. 그러나 Authentication 을 이용하기 위해서는 JEUS Web Server 에서의 설정뿐 아니라, 실제 사용자의 사용자명과 패스워드를 만들어서 저장 하여야 할 필요가 있다.

JEUS Web Server 를 먼저 설치한 다음 JEUS Web Server 가 설치된 Directory 를 보면 실제 실행 파일들이 있는 `bin/` 디렉토리 아래에 wsmkpw 라는 실행 파일이 있는 것을 볼 수 있다. 이 `wsmkpw` 라는 실행 파일을 통하여 사용자명과 패스워드를 만들어 낼 수 있다. 실제 수행의 간단한 예를 보면 다음과 같다.

```
wsmkpw /usr/local/webserver/bin/test newuser
New Password: *****
Retype Password: *****
```

위와 같이 하면, `/usr/local/webserver/bin` 아래에 `test` 라는 파일이 생기고 `newuser` 라는 새로운 사용자가 저장된다. 그리고 내부에는 입력한 패스워드가 암호화 되어 저장된다.

이 파일을 이용하여 JEUS Web Server 는 실제 Authentication 을 수행하는 것이다.

JEUS Web Server 의 설정 작업 도중 AUTHENT 라는 항목을 기억할 것이다. 이 항목에서 UserFile 이라는 항목이 있는데 여기에 `test` 라는 파일을 등록하면 JEUS Web Server 가 Authentication 을 수행하는 과정에서 이 파일을 이용하여 Authentication 작업을 하게 된다.

앞에서 설명한 바와 같이 `wsmkpw` 라는 실행 파일을 이용하여 사용자명과 패스워드를 등록한 후, 자신의 원하는 항목에 이 것을 추가하면 사용자가 그

항목에 대한 요구를 하는 경우마다 JEUS Web Server 는 Authentication 작업을 수행할 것이다.

이러한 Authentication 작업은 Web 의 초창기에 많이 이용되었다. 간단한 응용으로 사용자의 인증 작업을 수행할 수 있었기 때문이다. 그러나, 이 Authentication 시에 사용하는 암호화 방법이 아주 일반적이고도 해독이 가능하기 때문에 전자 상거래를 구현한 Site 등에서는 많이 외면 되고 있다. 또한, 사용자의 패스워드가 Internet 으로 흘러 들어 가게 되면서 나쁜 의도를 가진 사람들이 이 패스워드를 중간에 capture 하여 해독하는 사례도 발생하면서 더 강력한 인증과 암호화 방법이 강구 되었다.

이러한 문제가 대두되는 가운데 Netscape 사에 의해서 Secure Socket Layer 라는 것이 이러한 문제를 대처하기 위하여 등장하였다. 이는 기존 암호화 방법을 한 차원 높게 만든 것으로 비록 누군가 패스워드나 사용자명을 가져갔다 하더라도 암호를 해독하는 것이 아주 어렵고 거의 불가능하기 때문에 보안에 완벽을 기할 수 있는 방법으로 인정 받고 있다..

JEUS Web Server 에서 Authentication 과 SSL 을 이용하기 위해서는 JEUS Web Server 의 환경 파일에 이를 이용하기 위한 별도의 설정이 필요하다.

먼저 Authentication 을 이용하기 위해서는 AUTHENT 절을 설정하여야 한다. 이는 JEUS Web Server 에서 보안을 위하여 설정할 Authentication 에 대한 정의를 내리는 절이라고 할 수 있다. 그리고 이 AUTHENT 절에서 선언한 것을 각각의 Server Group 에 선언을 해 주면 된다. 즉, 각 Server Group 단위로 Authentication 을 설정하는 것이 가능하다는 것이다. 만약 사용자가 CGI 에 대해서 Authentication 을 이용하고 싶다면, 이 CGI 를 선언한 Server Group 에 AUTHENT 절에서 선언한 Authentication 이름을 선언한다. 이에 대한 자세한 내용은 부록 F 의 SVRGROUP 절에서 AuthentName 절을 참조하기 바란다.

AUTHENT 절은 다음과 같은 구조를 가지고 있다.

```
*AUTHENT
 [AUTHENT_NAME]   [TYPE], [USERFILE [,GroupFile]]
```

AUTHENT 의 이름은 사용자가 원하는 이름을, TYPE 은 사용자가 이용하고 싶은 Encryption 방법을 적으면 된다. 일반적으로 이용되는 방법은 Basic 과 MD5, RSA 등이 있다. JEUS Web Server Version 3.0 에서는 Basic 에 대한 설정만을 지원하고 있다. 다른 기타 암호화 방법은 차후의 Version 에서 제공할 것이다. 그리고, USERFILE 은 위의 AUTHENT 절에서 설명한 wsmkpw 란 것을 이용하여 만들어진 Password 파일을 말하는 것이다.

위의 AUTHENT 절에서 설명한 방법으로 만들어진 Password 파일을 이 AUTHENT 에서 이용하여야 하기 때문에 그 파일의 위치를 적어야 한다. 물론, 당연히 그 파일은 먼저 존재하여야 한다.

이와 같은 설정 방법을 이용하여 간단한 AUTHENT 절에 대한 예제를 보이면 다음과 같다.

```
*AUTHENT
 authent1   Type = Basic,
                  UserFile = "/usr/local/webserver/bin/pwfile"
```

AUTHENT 절에 대한 자세한 설명은 부록 F 를 참조하기 바란다.

위와 같은 설정을 하였다면, 사용자가 원하는 설정으로 Authentication 을 이용할 기본적인 준비가 된 것이다. 이젠 이를 이용하여 원하는 작업에 이 AUTHENT 절에서 정의한 항목을 실제로 연결하기만 하면 된다. 이 때 사용자가 이용하는 JEUS Web Server 의 환경 변수는 AuthentName 이다. 이를 이용하여 AUTHENT 절에 선언한 Authentication 방법을 이용하게 되는 것이다. 간단한 예로 사용자가 CGI 작업에 Authentication 을 설정하고 싶다면 다음과 같은 방법으로 설정하면 된다. (authen1 은 위의 예에서 설정되었다.)

<<*http.m*>>

```
*SVRGROUP
cgig  NODENAME = webmain, SVRTYPE = CGI,
                  AuthentName = authent1
```

위와 같은 형식으로 설정하면 JEUS Web Server 는 사용자가 CGI Service 를 요청할 때마다 사용자에게 Authentication 작업을 수행할 것이다. 단, 이 때 주의할 것은 이 Authentication 작업이 Server Group 단위로 이루어 진다는 점이다. 따라서 이 CGI Group 을 설정하면 이에 대응하는 Server 들은 모두 Authentication 작업을 수행할 것이다. 따라서 만약 사용자가 특정 CGI 는 Authentication 을 수행하고 또 다른 CGI 는 아니라면 두개의 Server Group 을 설정하여야 한다. 이에 대한 예는 다음과 같다.

<<*http.m*>>

```
*SVRGROUP
cgig         NODENAME = webmain, SVRTYPE = CGI
cgig_authent NODENAME = webmain, SVRTYPE = CGI,
             AuthentName = authent1
*SERVER
cgi1         SVGNAME = cgig
cgi2         SVGNAME = cgig_authent
```

```
*URI
uri1        Uri = “/cgi-bin/”, SVRTYPE = CGI,
            SVRNAME = cgi1
uri2        Uri = “/cgi/”, SVRTYPE = CGI,
            SVRNAME = cgi2
*ALIAS
alias1      URI = “/cgi-bin/”,
            RealPath =“/usr/local/webserver/cgi-bin/”
alias2      URI = “/cgi/”,
            RealPath = “/usr/local/webserver/cgi/”
```

위와 같이 두 가지로 Group 을 설정한 후, Authentication 을 원하는 것은 cgig_authent 라는 Server Group 을 이용하여 Server 를 설정하고, 다른 것들은 cgig 를 그대로 이용하여 설정하면 된다. 이런 설정은 Server Group 절에 선언될 수 있는 모든 것에 적용된다.

8.5.2 SSL 의 설정

JEUS Web Server 에서 SSL 을 이용하기 위해서는 별도의 설정이 필요하다. 이는 여타 다른 Authentication 이나 Log 절의 선언과 마찬가지로, SSL 절을 선언하여 이를 이용하는 방식으로 진행된다.

JEUS Web Server 에서 SSL 을 이용하기 위해서는 먼저 NODE 절이나, 아니면 SSL 을 이용하고 싶은 Virtual Host 를 선언한 VHOST 절에서 SSLFLAG 를 선언하여야 한다. 기본적으로 JEUS Web Server 는 SSL 을 이용하도록 되어 있지 않다. 따라서, 반드시 SSLFLAG 를 설정하여야 SSL 을 이용할 수 있다. 이 SSLFLAG 는 Y 와 N 으로 설정하는 것이 가능하다. Y 라는 값을 기입하면, SSL 을 이용한다는 것이 되고, N 은 이와 반대의 역할을 한다. 그러나, SSLFLAG 에 대한 설정이 없으면, 기본적으로 SSL 을 이용하는 것이 아니기 때문에, N 은 거의 쓰일 일이 없을 것이다. 다음은 SSLFLAG 의 사용 예이다.

```
SSLFLAG = Y
```

위와 같이 SSLFLAG 를 Y 로 설정하여 SSL 을 이용하겠다고 선언한 후에는, 이제 SSL 에 대한 설정을 하여 주어야 한다. 이 SSL 절의 형태는 다음과 같다:

```
SSL
SSL NAME    Certificatefile, CertificateKeyFile,
            CACertificatePath, CACertificateFile,
            VerifyDepth, VerifyClient, RandomFile,
            ...
```

이에 대한 자세한 내용은 부록 F 의 SSL 절을 참조하기 바란다. 이곳에서는 기본적인 설정에 필요한 정보만 기술할 것이다. 그리고, 이 선언한 SSL 을 Authentication 절에서 하듯이 각각의 SSL 선언을 필요로 하는 절에 선언하면 된다. 단, 주의 할 점은 Authentication 은 SVRGROUP 절에서 선언하였으나 SSL 절은 Node 절이나 VHOST 절에서 SSL 을 이용하는 곳에서 선언하여야 한다. 선언하는 방법은 SSLNAME 라는 항목을 이용한다.즉, SSLNAME 라는 항목에 SSL 절에서 선언한 것을 적어주면 된다. 이에 대한 사용 예제는 다음과 같다.

```
SSLNAME = “ssl1”
```

그러면 이 절에서는 ssl1 에서 선언한 값을 토대로 SSL 을 이용하겠다는 것이 된다. 그리고 위에서 밝힌 바와 같이 ssl1 이라는 항목은 SSL 절에서 이루어진다. 이에 대한 예는 다음과 같다.

```
*SSL
ssl1  CertificateFile = "/user/webserver/ssl/newcert.pem",
      CertificateKeyFile = "/user/webserver/ssl/newcert.pem",
      RandomFile = "/user/webserver/bin/.rnd, 2048",
      RandomFilePerConnection = "/user/webserver/bin/.rnd, 512",
      VerifyClient = 0,
      VerifyDepth = 10,
      FakeBasicAuth = Y
```

이와 같이 ssl1 을 선언하였고 이를 실제 적용하는 방법에 대한 예는 다음과 같다. (이 방법은 Authentication 과 거의 같은 방식이다.)

<<*http.m*>>

```
*NODE
test  WEBTOBDIR=”/user/webserver”,
      SHMKEY=69000,
      HTH=1,
      PORT=80,
      SSLFLAG=Y,
      SSLNAME=”ssl1”,
      ...
```

위의 예를 보면, NODE 절에서 SSL 을 이용하기 위해서 SSLFLAG 라는 항목을 선언하고 이어서 SSLNAME 이라는 것을 선언하였음을 알 수 있다. 그리고 SSLNAME=”ssl1”이라는 항목에 의해 이 NODE 는 ssl1 에서 정의된 항목을 통해서 SSL Service 를 한다는 것을 알 수 있다.

우리는 앞의 설정파일의 개념이라는 절을 통해서 DOMAIN 이나 NODE 절에서 선언한 값을 그 이후에 선언되는 절에서 새로 재정의 할 수 있다는 것을 알았다. 그것은 이 SSL 에서도 유효하게 되는데, 이 때, Virtual Host 를 정의하면서 이와 같은 개념이 적용 된다. 즉, NODE 절에서 SSLFLAG 를 Y 로 선언하면 이후에 선언되는 Virtual Host 를 선언하는 VHOST 절에서 기본적으로 SSL 이 적용되게 된다. 따라서, 굳이 SSLFLAG=Y 라고 VHOST 절에서 선언을 하지 않아도 SSL 을 자동적으로 이용할 수 있게 된다는 것이다. 이 때 만약 사용자가 특정 VHOST 에서 SSL 을 이용하고 싶다면 NODE 절의 SSLFLAG = N 으로 하고 SSL 을 사용하고자 하는 VHOST 절의 SSLFLAG=Y 로 선언하여 SSL 기능을 추가하면 된다. 이는 그 이후의 값에는 영향을 미치지 않는다.

즉, NODE 절에서 선언한 값만이 그 이후에 영향을 미치게 되는 것이다. 그러므로 SSL 을 이용하게 되는 경우 이러한 상관 관계를 명확히 하여 설정을 하여야 한다.

9 JEUS 와의 연동

9.1 소개

JEUS Web Server 는 Tmax Soft 에서 제공하는 차세대 Web Application Server 인 JEUS 와 연동되어, 일반 Web Server 에서 제공하지 못하는 기능들을 구현할 수 있다. 일반적인 Web Service 를 포함한 전자상거래에서는 Security 와 Transaction 등이 반드시 보장되어야 한다. 이 기능은 JEUS Web Server 에서도 제공하지만, 대용량의 Data 를 다루는 일이나, 사용자의 관리 등의 측면에서는 Web Application Server 인 JEUS 를 이용하는 것이 바람직하다. 사용자가 대규모의 전자상거래 시스템을 구축하려 한다면 JEUS Web Server 와 JEUS 를 연동하여 이용하는 것이 가장 바람직할 것이다

9.2 JEUS Web Server 와 JEUS 연동 개요

JEUS Web Server 와 같은 Web Server 에서는 HTML, CGI, SSI 등의 Service 들을 제공한다. 물론 이런 Service 를 가지고 간단한 전자상거래 및 기타 사업들을 충분히 구현가능하나 많은 Data 를 처리하거나 Transaction 및 DataBase 에 대한 연결들을 필요로 하는 것에는 JEUS Web Server 만의 기능으로는 부족하다고 할 수 있다.

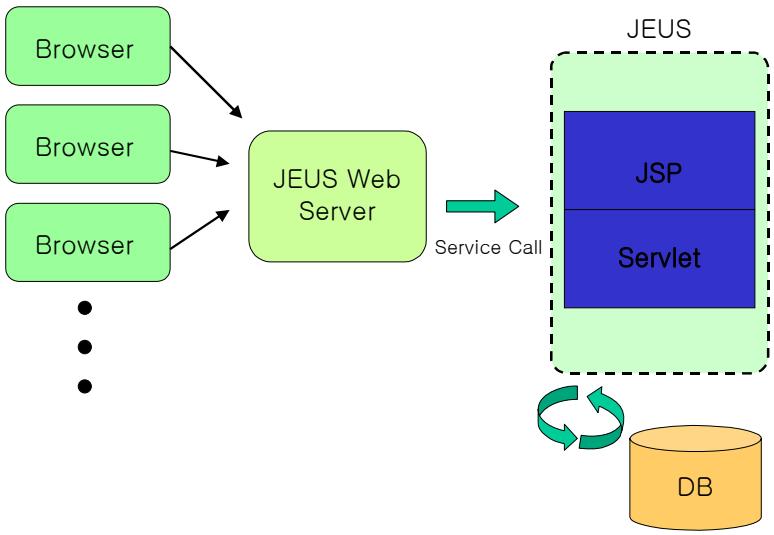

그림 6. JEUS Web Server 와 JEUS 연동

이에, JEUS 라는 Web Application Server 를 이용하여 JEUS Web Server 에서 미처 제공하지 못하는 기능들을 이용할 수 있다.

JEUS Web Server 와 JEUS 의 연동은 JEUS Web Server 의 설정파일에 JEUS 와 연동할 부분을 설정하여 주면 된다. 이 연동을 위한 방법을 설명하면 다음과 같다.

먼저 설정파일에서 JEUS 와 연결을 맺는 Port (설정파일의 Node 절에 JSVPort 항목) 번호를 설정하여 준다. 이 Port 번호는 기본적으로 9999 로 설정하고, JEUS 에서도 이 값으로 설정한다. 그러나, 만약 사용자가 다른 Port 를 이용하고 싶다면 이 Configuration 항목을 설정하면 된다. 마찬가지로 JEUS 에서 JEUS Web Server 와 연결하는 부분을 설정하면 된다. JEUS 에서는 설정파일로 xml 파일을 이용하는데 이 곳에 JEUS Web Server 와 연동을 위한 Port 가 있다. 이것을 JEUS Web Server 에서 설정한 것과 같은 값으로 설정하면 된다.

JEUS Web Server 에서 Port 설정이 끝나면 JEUS 와 연동준비가 끝난 셈이다. 이제 JEUS 를 Booting 하면 된다. 그러면 JEUS 의 웹 서버로서 JEUS Web Server 가 Booting 된다 . 이 때 중요한 것이 Booting 하는 순서이다. 현재 구조상 JEUS Web Server 가 JEUS 에서 연결을 맺으려 하는 것을 Listen 하는 상태이므로,먼저 JEUS Web Server 를 Booting 시키던지 ws_engine 으로써 JEUSMain.xml 환경 파일에 설정하여 JEUS 를 부트 시키면서 같이 JEUS Web Server 를 기동 시킬 수 있다. 이렇게 JEUSMain.xml 환경 파일에 JEUS Web Server 등록하여 기동 시킬 때는 ws_engine 을 Servlet Engine 보다 먼저 설정해 주어야 한다. 그리고 <sequential-start>태그 부분을 true 로 설정해 주

어야 한다. 그래야 등록된 엔진 순서대로 Booting 된다. JEUS Web Server 가 Booting 되어 Servlet Engine 과 연결을 위한 Port 에서 Listen 상태로 되어 있고, Servlet Engine 이 Booting 되면서 JEUS Web Server 웹 서버와 연결을 맺는다. 연결이 성공하면 연결이 성공되었다는 Message 가 나오게 되고, 이 후에 Servlet 이나 JSP 등 JEUS 의 Servlet Engine 을 이용해야 하는 사용자의 요구는 모두 이 Port 를 통해서 JEUS 에 전달되어 처리된다.

JEUS Web Server 는 JEUS 웹 어플리케이션 서버의 default Web Server 로서 기존의 모든 Web Server 의 처리를 담당해 준다. 즉 정적인 페이지의 전송, CGI, SSI, PHP 등 일반적인 Web Server 의 작업을 처리해 주며 Servlet 이나 JSP 등의 요청이 왔을 때 JEUS 웹 컨테이너로 요청을 넘겨 주게 되고 결과를 받아 클라이언트 측으로 전송해 준다.

여기서 주의할 점은 내장 Web Server 나 Apache 의 연결과는 다르게 JEUS Web Server 의 경우 반드시 JEUS Web Server 가 먼저 기동이 되어 있어야 한다. 그것은 연결을 처음 설정하는 과정에서 JEUS Web Server 가 기다리는 역할을 하기 때문이다. 하지만 연결이 설정된 이후에는 다른 것과 마찬가지로 JEUS Web Server 가 웹 컨테이너의 클라이언트 역할을 해서 요청을 넘겨 주고 결과를 받는다.

또한 JEUS Web Server 와 JEUS 웹 컨테이너와의 연결은 persistent connection 이므로 중간에 네트워크 오류나 서버의 문제가 생기지 않는 이상은 연결을 끊지 않고 재사용한다.

JEUS Web Server 와의 연결 설정을 위해서는 두 가지 설정 파일이 필요하다. 하나는 JEUS Web Server 의 설정 파일인 `ws_engine.m` 이고 또 다른 하나는 웹 컨테이너의 설정 파일인 WEBMain.xml 이다.

일반적으로 JEUS Web Server 는 설정 파일인 ws_engine.m 이 /$JEUS_WSDIR/config/ 경로에 존재하고 JEUS 의 경우는 서블릿 엔진을 띄우기 위한 환경 파일인 WEBMain.xml 은 아래의 경로에 존재한다.

```
/($JEUS_HOME)/config/(nodename)/(nodename)_servlet_(enginename)/
```

디렉토리명은 JEUSMain.xml 파일 설정시 정해주었던 값들과 연관이 있는데, nodename 은 <node>태그의 <name>노드명</name>에서 지정한 값, enginename 은 <engine-container>태그의 <name>엔진명</name>에서 지정한 값과 동일해야 한다. 즉, 예를 들어 JEUS 가 설치된 디렉토리가 /user/tmaxsoft 이고, 노드명은 tmaxhost, 서블릿 엔진명은 engine1 이라면 WEBMain.xml 의 경로는 /user/tmaxsoft/config/tmaxhost/tmaxhost_servlet_engine1 이 된다.

위의 경로에 저장하는 이유는 JEUS 구동시, 적절한 웹 컨테이너 환경 파일(WEBMain.xml)을 찾을 수 있도록 하기 위함이다.

9.2.1 예제

부록 D 를 참조하기 바란다.

9.2.2 참조

JEUS Web Container 안내서.

10 결론

이것으로 JEUS Web Server 에 대한, 환경설정, 튜닝, 제어, 그리고 관리에 대한 설명을 마치도록 한다.

나머지 매뉴얼에서는 JEUS Web Server 에서 사용되는 중요한 툴인, wsadmin, wscfl, wsmkpw 그리고, 환경설정의 예제를 중점적으로 설명하고자 한다.

이 매뉴얼에서 사용된 JEUS Web Server 환경파일은 부록 F 를 참조하기 바란다.

A wsadmin: JEUS Web Server 관리툴 레퍼런스

A.1 소개

이 부록에서는 JEUS Web Server 시스템을 관리하기 위해서 사용하는 관리 툴인 wsadmin 의 사용목적 및 사용법에 대해서 설명할 것이다.

A.2 사용목적

JEUS Web Server 시스템이 동작하고 있다면 시스템의 관리가 필요하다. 예를 들어, JEUS Web Server 시스템의 종료 및 기동의 절차를 거치지 않고도 현재 환경 설정에 대한 정보를 확인해 보고 이를 동적으로 변경할 필요가 있다.

또한 제공되는 Service 들 중 특정 Service 에 대하여 현재 처리 상태 즉, 몇 건의 Service 를 처리해 왔으며, 평균 처리 시간이 얼마나 되며, 몇 건의 요청이 대기하고 있는지, 그 Service 를 제공받기 위해서는 얼마나 기다려야 하는지 등의 정보를 확인할 필요가 있다.

이 정보에 따라 그 Service 를 제공하는 Server Process 를 더 기동 시켜 준다든지 아니면 필요 이상의 Server Process 가 동작 중일 경우 몇 개는 종료 시키는 등의 조치도 필요하다. 이러한 동적인 시스템 관리를 위하여 JEUS Web Server 시스템에서는 wsadmin 이라는 프로그램을 제공한다.

A.3 실행방법

wsadmin 프로그램은 UNIX 환경의 shell 과 비슷한 Command Interpreter 이다. 즉, 항상 프롬프트 상태로 대기중이다가 입력되는 명령어를 해석하여 이를 실행하게 된다. 여러 Node 를 한 Domain 으로 사용하는 경우 wsadmin 으로 전체를 중앙 관리가 가능하며 각 Node 별로 로컬에서만도 관리가 가능하다.

wsadmin 프로그램은 다음과 같이 실행한다:

```
wsadmin [-l]
```

옵션

`-l`

wsracd 를 통하여 여러 Node 로 구성된 시스템을 중앙에서 집중 관리할 경우에 특별히 지정된 Node 만을 관리하기 위해서 사용하는 옵션이다.

A.4 wsadmin 명령어

wsadmin 툴에서는 아래의 명령어들을 통하여 JEUS Web Server 시스템을 기동 시키거나 종료 시킬 수 있으며, 현재 동작중인 시스템의 환경설정 내용을 조회 및 수정할 수 있다. 또한 각 Server Process 의 상태와 Service 의 처리상태 등을 확인할 수 있다:

`wi`

현재 접속된 JEUS Web Server 시스템의 환경정보를 알려 준다. 시스템 버전(Version), 사용 만료기한 정보(데모버전 사용 시) 등을 확인할 수 있다.

```
JEUS Web Server System Info: REAL version 3.0.11:
         maxuser = unlimited,
         node_count = 1,
         svgrpcount = 4,
         svr_count = 4, svc_alloc_count = 64
         rout_groupcount = 0, rout_elemcount = 0
         cousin_groupcount = 0, cousin_elemcount = 0
         backup_groupcount = 0, backup_elemcount = 0
JEUS Web Server All Node Info: node_count = 1:
-------------------------------------------------------------
no  name     nodeport   racport   shmkey   shmsize    hth
-------------------------------------------------------------
0   webmain  7777       3333      59000    22716      1
```

`ci`

현재 접속된 Client 의 환경 정보를 알려 준다. 현재 상태(status), 접속 IP Address, 처리건수(count)와 같은 Client 의 제반 정보를 확인할 수 있다.

```
$$8 webmain (wsadm): ci
HTH 0:
-------------------------------------------------------------
cli_id    status   count  lastin_time  ipaddr       usrname
-------------------------------------------------------------
  0       RDY      2      4            192.168.63.2
-------------------------------------------------------------
Total Connected Clients = 1
-------------------------------------------------------------
-
```

si

현재 동작중인 각 Server 의 정보를 알려 준다. 즉, 현재 상태(status, RDY: Service 가능, NRDY: Service 불능), 처리 건수(count), 큐잉 건수(qcount), 큐에서 삭제된 건수(qpcount), 최대 큐잉 건수(maxqcount)를 초과하여 반환된 건수(emcount), 해당 서버프로세스가 Restart 된 횟수(rscount), 해당 서버프로세스가 MaxRequests 로 설정된 값에 의해서 Reboot 된 횟수(rbcount)등을 확인할 수 있다.

```
$$9 webmain (wsadm): si
-------------------------------------------------------------
hth  svrname (svri)  status  count  qcount   qpcount
emcount rscount rbcount
-------------------------------------------------------------
 0   html (  0)   RDY   2     0     0     0     0    0
 0   cgi  (  1)   RDY   0     0     0     0     0    0
 0   ssi  (  2)   RDY   0     0     0     0     0    0
 0   jsv1 (  3)   NRDY  15    0     0     0     0    0
```

history

shell 에서와 같이 명령어를 기억함으로써 편리하게 사용할 수 있다.

```
$$15 webmain (wsadm): history
4: si
3: ci
```

```
2: st -p
1: st -s
0: ci
```

!

직전 명령어를 반복하며 history 에서 출력된 번호를 사용할 수도 있다.

```
$$21 webmain (wsadm): !5
si
------------------------------------------------------------
hth  svrname (svri)  status  count  qcount   qpcount
emcount rscount rbcount
------------------------------------------------------------
 0   html (  0)   RDY   2     0     0     0     0    0
 0   cgi  (  1)   RDY   0     0     0     0     0    0
 0   ssi  (  2)   RDY   0     0     0     0     0    0
 0   jsv1 (  3)   NRDY  25    0     0     0     0    0
$$21 webmain (wsadm): !
si
------------------------------------------------------------
hth  svrname (svri)  status  count  qcount   qpcount
emcount rscount rbcount
------------------------------------------------------------
 0   html (  0)   RDY   2     0     0     0     0    0
 0   cgi  (  1)   RDY   0     0     0     0     0    0
 0   ssi  (  2)   RDY   0     0     0     0     0    0
 0   jsv1 (  3)   NRDY  30    0     0     0     0    0
```

`config (cfg) [-d] [-n <node name>] [-g server group name] [-v server name]`

현재 동작중인 시스템의 환경정보를 알려 준다. 즉, 환경파일에서 정의된 Domain, Node, Server Group, Server, Service 별로 default 값까지 포함한 모든 환경정보를 확인할 수 있다.

config 명령어에는 다음과 같은 옵션이 있다.

옵션	내용

-d	Domain
-n <Node name>	Node
-g <Server Group name>	Server Group
-v <Server name>	Server

`stat(st) [-p <Server Process name>] [-s <Service name>] [–l] [-n <Number of Lines>]`

실질적인 시스템 동작 상태를 나타내며, 동작중인 Server Process 와 Service 에 대한 정보를 알 수 있다. 즉 Server Process 의 현재 상태 처리중인 Service 이름, 처리한 Service 개수, Service 에 대한 상태, Service 큐에 존재하는 Service 요청 개수등과 같은 동적인 정보를 확인할 수 있다.

stat 명령어의 약어는 'st'이며, 다음과 같은 옵션들이 제공된다:

옵션	내용
-p <Server Process name>	Server.
-s <Service name>	Service.
-l	Local mode (Local mode).
-n <Number of Lines>	Display 'n' number of lines.

`suspend [-v <Server Process name>]`

응용 Server 프로그램 오류 등으로 더 이상의 업무 처리가 불가능하여 이를 해결하기 위해 동작중인 Server Process 를 중지시킬 경우가 발생할 수 있다. 이와 같이 원인을 알 수 없는 문제로 인하여 Service 처리를 중지시키고 추가적인 작업을 필요로 하는 경우에 유용하게 사용할 수 있는 기능이다.

중지된 Server Process 는 현재 처리중인 Service 를 정상 완료한 후 더 이상의 동작은 중지하고, 큐에 있는 서비스들은 대기 상태가 된다. 이 때 계속적으로 요청되는 Service 들은 모두 큐에 적체된다.

Suspend 명령어에는 다음과 같은 옵션들이 제공된다:

옵션	내용
-v <Server Process name>	Server

`resume [-v <Server Process name>]`

동작 중지된 Server Process 의 활동을 재개 시킨다. 활동이 재개된 Server Process 는 큐에 대기중이던 Service 를 처리하기 시작하며 요청되는 Service 에 대해 처리 가능 상태가 된다.

resume 명령어에는 다음과 같은 옵션들이 제공된다:

옵션	내용
-v <Server Process name>	Server

`qp(Queue Purge) [-v <Server Process name>]`

업무의 폭주 현상이 발생하여 정상적으로 거래를 처리하지 못하는 경우, 현재 큐에 적체되어 있는 Service 요청을 삭제함으로써 원활한 업무수행을 유도하는 기능이다. 하루에도 수십만 건의 업무를 처리하는 은행이나 관공서에서 유용하게 사용될 수 있는 기능으로서 삭제된 업무는 웹 브라우저의 재요청을 통하여 다시 처리될 수 있다. JEUS Web Server 에서는 Server Process 별로 큐를 관리함으로써 관리자는 특정 Server 별로 큐를 삭제할 수 있다. 따라서 특정 업무별로 처리가 가능하며 이는 타 업무의 효과적인 수행에도 도움을 줄 수 있다.

```
$$1 webmain (wsadm): qp –v sdlsel
```

qp 는 Server 단위로 가능하며 다음과 같은 옵션들이 제공된다:

옵션	내용
-v <Server Process name>	Server

qp 를 통해 삭제된 Service 는 Web Browser 에게 에러를 반환하게 되고 Web Browser 에서는 "503 Service Temporarily Unavailable"이라는 에러를 화면에 나타낸다. 따라서 Web Browser 는 이에 맞게 적절한 대처를 할 수 있다.

`stat(Repeat Mode)`

현재 상태정보를 일정한 시간 간격을 두고 계속 모니터링 하고자 하는 경우에 사용하는 기능으로서 명령어는 다음과 같다.

```
$$1 webmain (wsadm): st -s -i 5 -k 30
$$1 webmain (wsadm): st -p -i 5 -f 30
```

첫번째 명령어는 5 초간의 간격을 두고 'st –s'를 30 번 수행하라는 의미이며 두번째 명령어는 5 초간의 간격을 두고 'st –p'를 30 초간 수행하라는 의미이다. 반복적인 명령어 수행은 상태 정보를 모니터링 할 뿐 아니라 업무수행에 대한 디버깅에도 많은 도움을 줄 수 있다. '**-i'** 다음에 Interval 을 설정하지 않으면 0 초로 동작한다.

여러 Node 가 한 Domain 으로 설정된 경우 또는 다수의 Service 와 Server Process 가 존재하는 경우, 한 번에 상태정보를 관리하기란 쉽지 않다. 따라서 로컬에 있는 Service 와 Server Process 만을 보기 위한 '**-l'** 옵션을 제공하며 '**-n'** 으로 적절한 라인을 설정할 수도 있다**.**

```
$$1 webmain (wsadm): st -s -l -n 24
$$1 webmain (wsadm): st -p -l -n 24
```

```
set (-d <variable> bt | tt | nn) |
     (-g <variable> ld) |
     (-v <variable> mq | aq | fc | mr) |
     (-s <variable> pr | st)
```

현재 설정되어 있는 환경파일의 설정 값을 동적으로 변경할 수 있는 명령어로서 사용법은 다음과 같다. 변경 가능 항목은 wsadmin 에서 cfg 명령어를 통해서 확인할 수 있다. 각 항목 중에서 괄호로서 약어를 표시한 항목이 동적으로 변경 가능한 항목이다.

동적설정 변경 가능 항목의 예:

Set	Item
-d	bt (block time)
	tt (tx time)
	ni (nliveing)
-g	ld (load)
-v	mq (max qcount)
	aq (as qcount)
	fc (flowcontrol)
	mr (max restart)
-s	pr (prio)
	st (svc time)

예)

```
set –d[g,v,s] Domain[Server Group, Server, Service] Item Value
$$1 webmain (wsadm): set –d JEUS Web Server1 bt 300
$$1 webmain (wsadm): set –g svg1 ld 5
$$1 webmain (wsadm): set –v kfdl1 mq 1000
$$1 webmain (wsadm): set –s SYNC pr 100
```

```
ds(Disconnect Session) [-h <HTH number>]
      (n <Web browser name> | -i <idle time> |
      -c <client id> | -a) [-f]
```

현재 접속되어 있거나 아무 일도 수행하지 않는 Web Browser 를 강제로 연결 해제할 수 있다. Web Browser 정보를 얻을 수 있는 'ci' 명령어로 확인 후 사용한다.

옵션	내용
-h <HTH number>	연결을 끊고자 하는 HTH 를 설정
-n <browser name>	연결 시 설정된 웹 브라우저 이름.
-i <idle time>	지정된 시간(초)이상 경과된 웹 브라우저
-c <client ID>	웹 브라우저에 부여된 ID 번호.
-f	즉시 연결해제.
-a	해당 HTH 에 연결된 웹 브라우저 접속해제.

예)

```
ds [-h HTHno] -a | -n | -i xx | -c clid [-f]
ds -h0 -a
ds -c 1
ds -i 15 -f
ds -h2 -a –f
```

`rbs (Reboot Server Process) <new file> <old file>`

현재 사용 중인 Server Process 를 새로운 Process 로 변경하고자 하는 경우에 사용한다. WEBTOB_BKAPPDIR 를 환경 변수에 지정하고 새로운 프로그램을 지정된 Directory 에 옮기고 다음의 명령어를 수행한다:

```
$$1 webmain (wsadm): rbs <new file> <old file>
```

옵션	내용
new file	변경된 파일 (object file).
old file	현재 실행중인 파일 (object file).

`logstart <filename> / logend`

관리자는 Administration Tool 을 사용하여 여러 가지 실시간 정보를 살펴볼 수 있다. 따라서 주어진 상황에 맞는 즉각적이고 효과적인 조치를 취할 수 있다. 또한 Administration Tool 에서는 통계적 정보 분석을 위해 Administration Tool 정보 데이터를 로그로 남길 수 있다. Logging 파일은 현재 Directory 에 만들어진다. 따라서 Windows 의 경우에는 항상 wsadmin 이 실행되는 WEBTOBDIR/bin 에 생성된다.

로그 데이터를 통해 업무 폭주 시간, 불필요한 Server Process, 큐잉 상태 등을 확인하여 전반적인 시스템 분석이 가능하다. 명령어 사용은 다음과 같다.

예)

```
$$1 webmain (wsadm): logstart filename
$$1 webmain (wsadm): logend
```

logstart 명령어를 통해서 log 처리를 시작할 수 있으며 logend 명령어를 통해 끝낸다. logend 명령을 사용하지 않고 wsadmin 을 종료하는 경우에는, logstart 명령어로 지정한 파일에 log 기록을 남긴 후 종료하도록 되어 있으나, 특정 시스템에 경우에는 기록되지 않는 경우도 있으므로, 반드시 logend 명령어를 사용하여 종료할 것을 권장한다.

B wscfl: JEUS Web Server 컴파일 툴 레퍼런스

B.1 소개

이 부록에서는 JEUS Web Server 의 환경파일을 컴파일 할 때 사용하는 컴파일 툴인 wscfl 의 사용목적 및 사용법에 대해서 설명할 것이다.

B.2 사용목적

사용자가 이해할 수 있는 text 형식으로 된 문서를 정해진 문법에 맞추어 작성하고, 이를 기계가 인식할 수 있는 이진 파일로 변환하는 것이며, 이 과정 중에 문법적 오류나 시스템 설정 오류를 검증할 수 있는 것이다.

B.3 실행방법

컴파일 작업은 아래와 같이 wscfl 명령에 의해 이루어 진다:

```
wscfl [-v] |
      [-i <config_file_name>] [-o <config_binary_file_name>]
```

옵션:

`-i`

컴파일 대상이 되는 환경파일 이름을 지정한다.

`-o`

컴파일을 통해 만들어질 이진(Binary) 환경파일 이름을 지정한다. 지정하지 않으면 default 로 'wsconfig'라는 이름의 파일이 만들어진다.

`-v`

JEUS Web Server Version 을 확인한다.

C wsmkpw: JEUS Web Server 인증 툴 레퍼런스

C.1 소개

이 부록에서는 JEUS Web Server 에서 인증에 사용할 암호화된 파일을 생성하는 wsmkpw 툴의 사용목적 및 사용법에 대해서 설명할 것이다

C.2 사용목적

wsmkpw 라는 실행 파일을 이용하여 사용자명과 패스워드를 등록한 후, 자신의 원하는 항목에 이 것을 추가하면 사용자가 그 항목에 대한 요구를 하는 경우마다 JEUS Web Server 는 Authentication 작업을 수행할 것이다.

C.3 실행방법

wsmkpw 툴은 다음과 같이 실행한다:

```
wsmkpw <file_name_path> <user_name>
```

옵션:

`file_name`

인증파일이 있는 전체 경로

`user_name`

인증에 사용될 사용자 이름

D JEUS Web Server 환경설정 예

D.1 환경파일 예

D.1.1 JEUS Web Server 환경파일 1

<<*http.m*>>

```
*DOMAIN
JEUS_Web_Server1        NLiveInq = 30

*NODE
webmain     WEBTOBDIR = "/usr/local/webserver",
            SHMKEY = 69000,
            DOCROOT = "/usr/local/webserver/docs",
            HOSTNAME = "webmain.tmax.co.kr",
            PORT = "8080",
            JSVPORT = 9099,
            User = "nobody",
            Group = "nobody",
            Hth = 1,
            NodePort = 8888,
            RacPort = 3333,
            CliChkIntval = 60,
            CacheSize = 128,
            CacheEntry = 128,
            KeepAliveTimeOut = 30,
            KeepAliveMax = 100,
            IndexName = "index.html",
            Options = "-Index",
            LOGGING = "log1",
            ERRORLOG = "log2",
            SysLogDir = “/usr/local/webserver/log/syslog",
            UsrLogDir = “/usr/local/webserver/log/usrlog",
            UserDir = “/usr/local/webserver",
            EnvFile = webserver_env,
            DirIndex = "Index",
```

```
            Method = "GET, POST, HEAD",
            LanguagePrio = "kr",
             Listen = "192.168.63.2:8080,192.1.1.9:8080",
*VHOST
vhost1      DOCROOT="/usr/local/webserver/docs/vhost_docs",
            NODENAME = webmain,
            HOSTNAME = "test.tmax.co.kr",
            PORT = “8090”,
            User = "nobody",
            Group = "nobody",
            UsrLogDir = “/usr/local/webserver/vhost/log/usrlog",
            UserDir = “/usr/local/webserver/vhost",
            EnvFile = webserver_env,
            IndexName = "test.html",
            LOGGING = "log1",
            ERRORLOG = "log2"
*SVRGROUP
htmlg       NODENAME = webmain, SVRTYPE = HTML
cgig        NODENAME = webmain, SVRTYPE = CGI
jsvg        NODENAME = webmain, SVRTYPE = JSV
ssig        NODENAME = webmain, SVRTYPE = SSI
*SERVER
html        SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi         SVGNAME = cgig, MinProc = 1, MaxProc = 5
ssi         SVGNAME = ssig, MinProc = 1, MaxProc =2
jsv1        SVGNAME = jsvg, MinProc = 1, MAXProc = 10
*DIRECTORY
dir_test    DIRECTORY = "/usr/local/webserver/docs/vhost_docs",
            ForceMimetype = "CGI"
*URI
uri1        Uri = "/cgi-bin/", SVRTYPE = CGI
uri2        Uri = "/jsv/", SVRTYPE = JSV
*ALIAS
alias1      URI = "/cgi-bin/",
             REalPath = "/usr/local/webserver/cgi-bin/"
*DIRINDEX
Index       Options = "FANCY"

*LOGGING
```

```
log1        Format = "DEFAULT",
             FileName = "/usr/local/webserver/log/access.log",
            Option = "sync"
log2        Format = "ERROR",
             FileName = "/usr/local/webserver/log/error.log",
            Option = "sync"
*AUTHENT
authent1    Type = Basic,
            UserFile = "/usr/local/webserver/bin/pwfile"
*EXT
htm         MimeType = "text/html", SVRTYPE = HTML
any         MimeType = "text/html", SVRTYPE = CGI
jsp         MimeType = "test/jsp", SVRTYPE = JSV
```

D.1.2 JEUS Web Server 환경파일 2

<<http.m>>

```
*DOMAIN
JEUS_Web_Server1
*NODE
webmain     WEBTOBDIR = "/usr/local/webserver",
            SHMKEY = 69000,
            DOCROOT = "/usr/local/webserver/docs",
            PORT = "8080",
            JSVPORT = 9099,
            LOGGING = "log1",
            ERRORLOG = "log2"
*VHOST
vhost       DOCROOT = "/usr/local/webserver/docs/vhost_docs",
            NODENAME = webmain,
            HOSTNAME = "test.tmax.co.kr",
            PORT = “8090”,
            LOGGING = "log1",
            ERRORLOG = "log2"
*SVRGROUP
htmlg       NODENAME = webmain, SVRTYPE = HTML
cgig        NODENAME = webmain, SVRTYPE = CGI
jsvg        NODENAME = webmain, SVRTYPE = JSV
ssig        NODENAME = webmain, SVRTYPE = SSI
*SERVER
```

```
html        SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi         SVGNAME = cgig, MinProc = 1, MaxProc = 5, SvrCPC = 4
ssi         SVGNAME = ssig, MinProc = 1, MaxProc =2
jsv1        SVGNAME = jsvg, MinProc = 1, MAXProc = 10
*URI
uri1        Uri = "/cgi-bin/", SvrName = cgi, SVRTYPE = CGI
uri2        Uri = "/jsv/", SvrName = jsv1, SVRTYPE = JSV
*ALIAS
alias1      URI = "/cgi-bin/",
           RealPath = "/usr/local/webserver/cgi-bin/"
*LOGGING
log1        Format = "DEFAULT",
           FileName = "/usr/local/webserver/log/access.log",
            Option = "sync"
log2        Format = "ERROR",
           FileName = "/usr/local/webserver/log/error.log",
            Option = "sync"
*AUTHENT
authent1    Type = Basic,
            UserFile = "/usr/local/webserver/bin/pwfile"
*EXT
htm         MimeType = "text/html", SvrName = html,
              SVRTYPE = HTML
jsp         MimeType = "text/jsp", SvrName = jsv1,
              SVRTYPE = JSV
```

D.2 고급 환경 설정

D.2.1 CGI 예제

단순히 보여주기 위한 HTML 에서 진일보한 형태인 CGI 의 동작과정은 다음과 같다. HTML 문서 중 사용자에 대해 동적으로 반응해야 하는 부분이 있다면, 이 부분은 마치 C 언어의 함수처럼 특정한 방식으로 응용 프로그램을 호출하게 된다. 이 응용 프로그램은 필요한 동작을 수행하고 그 결과를 다시 보내주면, 이 결과가 화면에 나타나 마치 웹 페이지가 원하는 요구를 알아서 처리해 준 것처럼 보이는 것이다.

게시판을 예로 들어보자. 우리가 글을 남길 수 있는 웹상의 게시판들은 우리가 쓴 글을 데이터로 받은 다음 이 데이터를 이용해 다시 HTML 문서를 만들

어 브라우저에게 전달해 준다. 즉, 게시판에 있어서 CGI 의 역할은 우리가 자판으로 치는 문자를 HTML 문서로 변환하는 역할을 하는 것이다. 브라우저는 이미 작성된 HTML 을 보여주는 역할만 하는 것이므로, 미리 작성하여 준 화면만을 보여줄 수 밖에 없는데 CGI 를 통해 우리는 이것에 실시간으로 접근하여 글을 남길 수 있는 것이다. 즉, 게시판 프로그램은 웹 페이지에 직접 쓸 수 있는 "연필" 과 같은 기능을 하는 것이다.

D.2.2 C 언어를 이용한 게시판 구현

유닉스 환경 하에서 C 를 이용한 게시판 작성을 예제로 하였다. 간단한 글과 주소를 남길 수 있는 이 웹 페이지는 board.html 에서 board.cgi 를 호출함으로써 구현한다.

순서는 아래와 같다:

1. 환경 파일인 `ws_engine.m` 을 시스템에 맞게 수정한다.
2. HTML 문서를 DOCROOT 로 옮긴다.
3. `board.html` 의 `ACTION="/cgi-bin/board.cgi"` 부분이 자신의 환경에서 `board.cgi` 가 있는 곳을 가리키도록 한다
4. `board.c` 및 `qDecoder.h`, `qDecoder.c` 를 cgi 가 구동 되는 Directory 로 옮긴다.
5. `board.c` 를 컴파일 한다. (cc 또는 gcc 가 깔려있어야 한다.)

   ```
   $ cc –o board.cgi board.c qDecoder.c, or
   $ gcc board.c qDecoder.c –o board.cgi  (board.cgi created)
   ```

6. 생성된 `board.cgi` 의 퍼미션을 755 로 해둔다.

   ```
   $ chmod 755 board.cgi
   ```

7. JEUS Web Server 를 기동한 후, 브라우저를 띄우고 게시판을 호출해 본다.

   ```
   http://192.168.63.2:8080/board.html
   ```

   (IP address 는 현재 JEUS Web Server 가 구동 되고 있는 machine 의 것을 쓴다.)

내용을 입력하고 "submit" 버튼을 누르면 `board.cgi` 가 호출되어 아래와 같이 게시판에 입력된 화면을 만들어 준다.

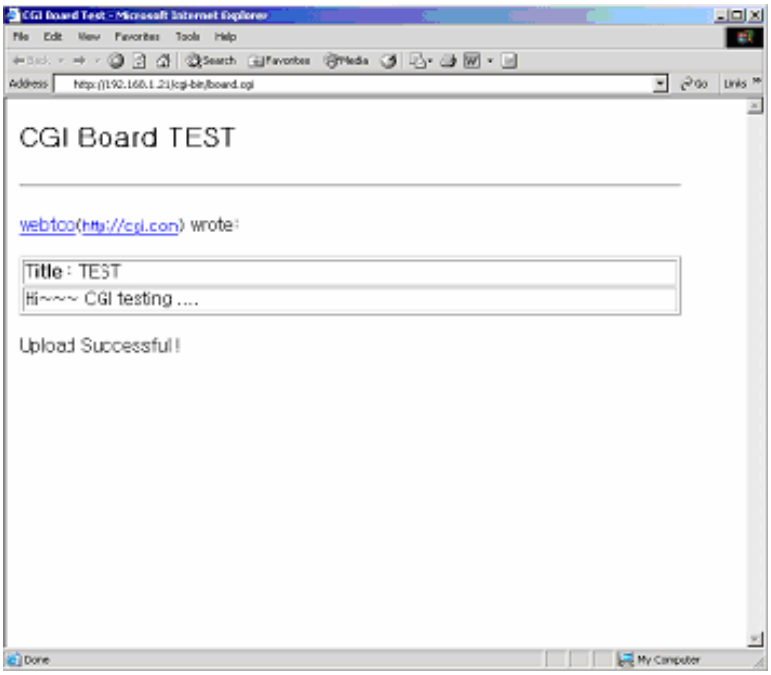

그림 7. board.cgi 스크린 샷

게시판을 만드는데 필요한 파일은 아래와 같다. 환경 파일인 `ws_engine.m` 과 HTML 문서인 `board.html`, 그리고 C 로 된 `board.c` 이다.

첨부 1 : 설정파일 (sample.m)

<<sample.m>>

```
*DOMAIN
JEUS_Web_Server1
*NODE
webmain            WEBTOBDIR = "/usr/local/webserver",
                   SHMKEY = 72000, HTH=1,
                   DOCROOT = "/usr/local/webserver/docs",
                   PORT = "8080"

*SVRGROUP
htmlg              NODENAME = webmain, SvrType = HTML
cgig               NODENAME = webmain, SvrType = CGI
*SERVER
html               SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi                SVGNAME = cgig,  MinProc = 1, MaxProc = 5

*URI
```

```
uri1              Uri = "/cgi-bin/", SvrName = cgi, Svrtype = CGI
*ALIAS
alias1            URI = "/cgi-bin/",
                  RealPath = "/usr/local/webserver/cgi-bin/"
```

첨부 2 : board.html

<<*board.html*>>

```
<HTML>
<HEAD> <TITLE>JEUS Web Server Board</TITLE> </HEAD>
<BODY>
  <H2><big>JEUS Web Server Board Upload</big></H2>
  <HR WIDTH=500 ALIGN=left> <BR>
  <FORM METHOD=post ACTION="/cgi-bin/board.cgi">
  <TABLE WIDTH=500 BORDER=0>
  <TR>
    <TD>Writer</TD>
    <TD><INPUT NAME=writer SIZE=20></TD>
  </TR>
  <TR>
    <TD>Title</TD>
    <TD><INPUT NAME=title SIZE=50></TD>
  </TR>
  <TR><TD COLSPAN=2><B>Contents</B></TD></TR>
  <TR><TD COLSPAN=2>
    <TEXTAREA NAME=doc COLS=60 ROWS=10></TEXTAREA>
  </TD> </TR>
  <TR>
    <TD>E-Mail</TD>
    <TD><INPUT NAME=email SIZE=40></TD>
  </TR>
  <TR>
    <TD>Home Page</TD>
    <TD><INPUT NAME=homepage SIZE=40 VALUE="http://"></TD>
  </TR>
  </TABLE>
  <BR>
  <INPUT TYPE=submit VALUE="    Submit    ">
  <INPUT TYPE=reset VALUE="     Clear     ">
  </FORM>
```

```
</BODY>
</HTML>
```

첨부 3: board.c

<<board.c>>

```
#include "qDecoder.h"
int strcheck(char *str) {
  /* Check whether the phrase is NULL or 0 in length */
  if (str == NULL) return 0;
  if (strlen(str) == 0) return 0;
  return 1;
}
int main(void) {
  char *name, *title, *doc;
  char *email, *homepage;
  /* Obtain input values */
  name = qValue("writer");
  title = qValue("title");
  doc = qValue("doc");
  email = qValue("email");
  homepage = qValue("homepage");
  /* Check whether input values are right */
  if (!strcheck(name)) qError("Type Your Name !");
  if (!strcheck(title)) qError("Type Title !");
  if (!strcheck(doc)) qError("Write Your Messages !");
  if (strcheck(email)) if (!qCheckEmail(email))
    qError("Type Your E-Mail !");
  if (strcheck(homepage)) if (!qCheckURL(homepage))
    qError("Type Your Homepage URL !");
  /* Here added to the bulletin board. */
  /* Output of processing result – Confirming input */
  qContentType("text/html");
  printf("<HTML>\n\n");
  printf("<HEAD> <TITLE> CGI Board TEST </TITLE> </HEAD>\n\n");
  printf("<BODY>\n");
  printf("<H2> CGI Board TEST </H2>\n");
  printf("<HR WIDTH=600 ALIGN=left>\n<BR>");
  if (strcheck(email)) printf("<A HREF=\"mailto:%s\">%s</A>",
email, name);
```

```
  else printf("%s", name);
  if (strcheck(homepage))

 printf("<SMALL>(<A HREF=\"%s\">%s</A>)</SMALL>", homepage,
homepage);
  printf("wrote<BR><BR>\n");
  printf("<TABLE WIDTH=600 BORDER=1>\n");
  printf("<TR><TD><B>Title</B> : %s</TD></TR>\n", title);
  printf("<TR><TD>%s</TD></TR>\n", doc);
  printf("</TABLE>\n\n<BR>Upload Successful !\n");
  printf("</BODY>\n\n</HTML>");
  qFree();
}
```

D.2.3 펄을 이용한 게시판 구현

Perl 이란 언어는 스크립팅 언어이다. 따라서 컴파일 과정이 필요 없이 바로 수행이 가능하기 때문에 프로그램의 작성과 디버깅이 매우 쉽다. 이런 이유로 스크립팅 언어가 선호된다. Perl 은 무료로 다운로드 가능하다. Perl 을 사용하기 위한 기초 준비작업은 여기서 다루지는 않겠다. 여기서는 JEUS Web Server 에서 Perl 을 기동하기 위한 환경파일과 Perl 로 구현한 방명록을 담았다.

Perl 을 기동할 수 있는 기본환경이 갖추어진 상태에서 다음 순서대로 해보자.

1. 환경 파일인 `ws_engine.m` 을 시스템에 맞게 수정한다.
2. `$which perl` 로 perl 의 path 를 알아놓는다.
3. `guestbook.cgi` 를 편집 창을 열어 시스템에 맞게 수정한다.
4. `guestbook.cgi` 를 cgi 가 구동 되는 Directory 로 옮기고 755 로 퍼미션을 준다.

   ```
   $chmod 755 guestbook.cgi
   ```

5. `guestbook.cgi` 의 데이터를 저장하기 위한 `book_data` 파일을 만들고 666 으로 퍼미션을 준다.

   ```
   $touch book_data
   $chmod 666 book_data
   ```

6. 마찬가지로 lock 디렉토리를 만들고 777 로 퍼미션을 준다.

```
$mkdir lock
$chmod 777 lock
```

7. 시스템을 기동한 후, 아래와 같이 테스트를 해본다.

8. 브라우저에서 , http://IP address:port/cgi-bin/guestbook.cgi

요구하는 정보를 간단히 쓴 다음 “send” 버튼을 눌러 방명록에 기록은 남겨 보자. 이제 guset 2 가 남긴 말 위에 JEUS Web Server 가 남긴 말이 추가될 것이다.

실행된 화면은 아래와 같다.

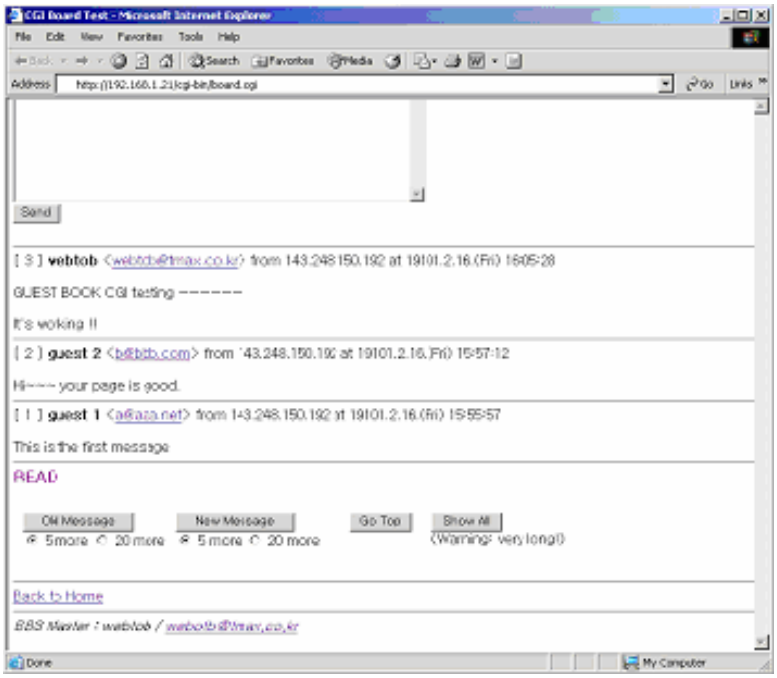

그림 8. guestbook.cgi 스크린 샷

첨부 1 : 설정파일 (sample.m)

<<sample.m>>

```
*DOMAIN
JEUS_Web_Server1
*NODE
webmain     WEBTOBDIR = "/usr/local/webserver",
            SHMKEY = 72000, HTH=1,
            DOCROOT = "/usr/local/webserver/docs",
```

```
            PORT = "9989"
*SVRGROUP
htmlg       NODENAME = webmain, SvrType = HTML
cgig        NODENAME = webmain, SvrType = CGI
*SERVER
html        SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi         SVGNAME = cgig,  MinProc = 1, MaxProc = 5
*URI
uri1        Uri = "/cgi-bin/", SvrName = cgi, Svrtype = CGI
*ALIAS
alias1      URI = "/cgi-bin/",
            RealPath = "/usr/local/webserver/docs/cgi-bin/"
```

첨부 2 : guestbook.cgi

<<guestbook.cgi>>

```
#!/usr/local/bin/perl
# standard URL (general homepage address)
$Base_href       = 'http://webmain.tmax.co.kr:9081/';
# path of this file based on $base_href
$Cgi_file        = 'cgi-bin/guestbook.cgi';
# lock directory path
$Lock_dir        = 'lock/guestbook';
# file where BBS data will be loaded
$Bbs_file        = 'book_data';
# title to be linked
$Link_name       = 'Back to Home';
# URL to be linked(based on $base_href)
$Link_url        = './';
# title
$Bbs_title       = 'Guest Book';
# manager's name
$Bbsmaster_name  = 'JEUS Web Server';
# manager's email address
$Bbsmaster_email = 'webotb@tmax.co.kr';
# maximum size of each message (byte) (for SPAM prevention)
$Max_msg_size    = 8000;
$Titlecolor      = '#800080';
$Bgcolor         = '#fafff8';
$Textcolor       = '#000000';
```

```
$Linkcolor       = '#401080';
$Bbs_write_title = 'WRITE';
$Bbs_read_title  = 'READ';
$Rem_name        = '';
$Rem_email       = '';
$Rem_msg         = '' .
                   '' .
                   '';
$Btn_write       = 'Send';
$Btn_read_old    = 'Old Message';
$Btn_read_new    = 'New Message';
$Rem_bbs_end     = 'End of Messages';
$Rem_bbs_back    = 'Go Top';
$Bbs_logo        = '';
$Bbs_logo_addr   = 'http://';
$Delimiter       = chr(30);
$Prev_num_skip   = 0;
##### main routine #####
&read_file;
&read_form;
if ($ENV{'REQUEST_METHOD'} eq "POST" && $Name ne "" && $Msg ne
"" ) {
        &remove_html_tags;              # remove HTML tags
        &new_msg;                         # make new BBS msg, use
&clock
        &save_msg;                        # save BBS msg
}
&show_header;                             # show header
&show_entry_form;                        # show BBS entry form
&show_bbs;                                 # show BBS msgs
&show_read_form;                          # show BBS read form
&show_footer;                             # show footer
exit;
##### sub routines #####
sub read_file {
        open (IN, "$Bbs_file") || &return_error (500, "File open
error", "Cannot access BBS file");
        @Bbs = <IN>;
        close (IN);
```

```
}
sub read_form {
        %Form = &read_input;
        $Name     = $Form{'name'};
        $Email    = $Form{'email'};
        $Msg      = $Form{'msg'};
        $Num_each = $Form{'num_each'};
        $Num_skip = $Form{'num_skip'};
}
sub remove_html_tags {
     $Name =~ s/\&/\&amp;/g;
     $Name =~ s/\</\&lt;/g;
     $Name =~ s/\>/\&gt;/g;
     $Email =~ s/\&/\&amp;/g;
     $Email =~ s/\</\&lt;/g;
     $Email =~ s/\>/\&gt;/g;
     $Msg =~ s/\&/\&amp;/g;
     $Msg =~ s/\</\&lt;/g;
     $Msg =~ s/\>/\&gt;/g;
     $Msg =~ s/\r\n/\n/g;              # Windows(CR,LF) -> LF
     $Msg =~ s/\r/\n/g;                 # Mac (CR)       -> LF
     $Msg =~ s/\n/<BR>/g;               # LF             -> <BR>
}
sub new_msg {
     my ($access_time) = &clock;       # get date and time
     my ($new_msg);
     my ($site) = ($ENV{'REMOTE_HOST'} ||$ENV{'REMOTE_ADDR'});
# delete msg larger than 4000 byte
     if (length($Msg) >= 4000) {
           $Msg = substr($Msg, 0, 4000);
             if (($Msg =~ tr/[\xA1-\xFE]//)%2 != 0) { chop
$Msg; }
        }
     if ($Email !~ /([\w\-]+\@[\w\-+\.]+[\w\-]+)/) {
           $Email = "";       # check valid email address format
        }
     $new_msg = "$Name"       . $Delimiter .       # name
                "$Email"     . $Delimiter .      # email address
                 "$site"      . $Delimiter .       # ip_address
```

```
                "$access_time" . $Delimiter .     # access time
                "$Msg"         . "\n";                # message
     @Bbs = ($new_msg, @Bbs);
}
sub save_msg {
     (&filelock ($Lock_dir) eq 'OK') || &return_error ("500",
"Lock error", "Too many access or cannot create lock file");
     open (OUT, ">$Bbs_file") || &return_error (500, "File write
error",  "Cannot access BBS file");
     print OUT @Bbs;
     close (OUT);
    (&fileunlock ($Lock_dir) eq 'OK') || &return_error ("500",
"Unlock error", "No lock file exist");
}
sub show_header {
        print "Content-type: text/html\n\n";
        print <<END_OF_HTML;
<HTML>
<HEAD>
<BASE HREF="$Base_href">
<TITLE>$bbs_title</TITLE>
</HEAD>
<BODY BGCOLOR="$Bgcolor" TEXT="$Textcolor" LINK="$Linkcolor">
<P ALIGN="CENTER"><FONT COLOR="$Titlecolor" SIZE="+2">
<B>$Bbs_title</B></FONT>
<br>
<div align="left"><A HREF="$Link_url">$Link_name</A></div>
<P ALIGN="RIGHT">
<HR>
END_OF_HTML
;
}
sub show_entry_form {
        print <<END_OF_HTML;
<a name="write"> </a>
<FONT COLOR="#800080" SIZE="+1">
<B>$Bbs_write_title</B></FONT>
<P>
<FORM ACTION=\"$Cgi_file\" METHOD=\"POST\">
```

```
<B>Name</B> $Rem_name
<BR><INPUT NAME="name" TYPE="text" SIZE = "40" MAXLENGTH="64">
<P>
<B>Email address</B> (optional) $Rem_email
<BR><INPUT NAME="email" TYPE="text" SIZE = "40" MAXLENGTH="72">
<P>
<B>Message</B> $Rem_msg
<BR>
<TEXTAREA COLS="64" ROWS="10" WRAP="VIRTUAL"
NAME="msg"></TEXTAREA>
<BR>
<INPUT TYPE="submit" VALUE="$Btn_write ">
</FORM>
<HR>
END_OF_HTML
;
}
sub show_read_form {
        print <<END_OF_HTML;
<font color="#800080" size="+1"><b>$Bbs_read_title</b></font>
<br><br>
<table border="0" cellspacing="10">
<tr valign="top">
<td>
<FORM ACTION=\"$Cgi_file\" METHOD=\"POST\">
<INPUT TYPE="submit" VALUE="$Btn_read_old">
<br>
<INPUT TYPE="radio" NAME="num_each" VALUE="5" CHECKED> 5 more
<INPUT TYPE="radio" NAME="num_each" VALUE="20"> 20 more
<INPUT TYPE="hidden" NAME="num_skip" VALUE="$Prev_num_skip">
</FORM>
</td>
<td>
<FORM ACTION=\"$Cgi_file\" METHOD=\"POST\">
<INPUT TYPE="submit" VALUE="$Btn_read_new">
<br>
<INPUT TYPE="radio" NAME="num_each" VALUE="-5" CHECKED> 5 more
<INPUT TYPE="radio" NAME="num_each" VALUE="-20"> 20 more
<INPUT TYPE="hidden" NAME="num_skip" VALUE="$Prev_num_skip">
```

```
</FORM>
</td>
<td></td>
<td></td>
<td>
<FORM ACTION=\"$Cgi_file\" METHOD=\"POST\">
<INPUT TYPE="hidden" NAME="num_each" VALUE="top">
<INPUT TYPE="submit" VALUE="Go Top">
</FORM>
</td>
<td></td>
<td>
<FORM ACTION=\"$Cgi_file\" METHOD=\"POST\">
<INPUT TYPE="hidden" NAME="num_each" VALUE="all">
<INPUT TYPE="submit" VALUE="Show All">
<br>
(Warning: very long!)
</FORM>
</td>
</tr>
</table>
<hr>
END_OF_HTML
;
}
sub show_bbs {
        local ($bbs_no, $serial_no, @show_bbs, @data,
$temp_email, $temp_msg);
        $bbs_no = $#Bbs + 1;
       unless (defined ($Num_each)) {$Num_each = 5;}
       unless (defined ($Num_skip)) {$Num_skip = 0;}
    if ($Num_each eq "all") { $Num_each = $bbs_no - $Num_skip; }
    if ($Num_each eq "top") { $Num_each = 5; $Num_skip = 0; }
    if ($Num_skip >= $bbs_no) { &show_end_of_bbs; }
    if ($Num_skip + $Num_each > $bbs_no) { $Num_each = $bbs_no -
$Num_skip; }
        if ($Num_each < 0 ) {
                $Num_skip = $Num_skip + $Num_each + $Num_each;
                $Num_each = 0 - $Num_each;
```

```
                if ($Num_skip < $Num_each) {
                        $Num_skip = 0;
                }
        } else {
                $Prev_num_skip = $Num_skip + $Num_each;
    # skip data at next loading
        }
        $serial_no = $bbs_no - $Num_skip;
        # first serial No.
        @show_bbs = splice(@Bbs, $Num_skip, $Num_each);
# displayed msgs
        while (@show_bbs) {
                @data = split (/$Delimiter/, shift(@show_bbs));
 # use slice of message file
    print "[ $serial_no ] <B>";
    print shift(@data);        # $Name
    print '</B>';
    if ($temp_email = shift(@data)) {       # $Email
    print " <<a href=\"mailto:$temp_email\">$temp_email<\/a>> ";
                }
    print ' from ';
    print shift(@data);           # $ip_address
    print ' at ';
    print shift(@data);           # $access_time
    print ' <P>';
    $temp_msg = shift(@data);
    $temp_msg =~ s/(http:\/\/)([\w\+\-\/\=\?\.\~\:\&\;\#]+)/
     <a href=\"$1$2\" target=\"_new">$1$2<\/a>/g;
    $temp_msg =~ s/([\w\-]+\@[\w\-+\.]+[\w\-]+)/<a
href=\"mailto:$1">$1<\/a>/g;
    print $temp_msg;              # $msg
    print "\n<HR>\n";
    $serial_no --;                # prepare next serial No.
        }
}
sub show_end_of_bbs {
        print "$Rem_bbs_end";
        print '<P>';
        print "<A HREF=\"$Cgi_file\">$Rem_bbs_back</A>";
```

```
        print "<HR>\n";
}
sub show_footer {
        print <<END_OF_HTML;
<A HREF="$Link_url">$Link_name</A>
<HR>
<I>BBS Master : $Bbsmaster_name
 / <A HREF="mailto:$Bbsmaster_email">$Bbsmaster_email</A></I>
<div align="right"><B><I><A
HREF="$Bbs_logo_addr">$Bbs_logo</A></I></B>
</div>
</body>
</html>
END_OF_HTML
;
}
#### other subroutines (shared with other perl cgi) ########
sub read_input {
        local ($buffer, @pairs, $name, $value, %FORM);
        if ($ENV{'REQUEST_METHOD'} =~ /^post$/i) {
                read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
        } else {
                $buffer = $ENV{'QUERY_STRING'};
        }
        @pairs = split(/&/, $buffer);
        foreach (@pairs) {
                ($name, $value) = split(/=/, $_);
                $value =~ tr/+/ /;
                $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",
hex($1))/eg;
                $FORM{$name} = $value;
        }
        return %FORM;
}
sub clock {
        my ($date);
        @days = ('Sun','Mon','Tue','Wed','Thr','Fri','Sat');
        ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
localtime(time);
```

```
        $mon ++;
        if ($hour < 10) { $hour = "0".$hour; }
        if ($min < 10) { $min = "0".$min; }
        if ($sec < 10) { $sec = "0".$sec; }
        $date = "19$year\.$mon\.$mday\.($days[$wday])
$hour\:$min\:$sec";
}
sub return_error {
        my ($status, $keyword, $msg) = @_;
        print "Content-type: text/html\n";
        print "Status: ", $status, " ", $keyword, "\n\n";
        print "<TITLE>CGI program Error</TITLE>\n";
        print "<H1>", $keyword, "</H1>\n";
        print $msg, "\n";
        print "<P>Please contact the webmaster for more
information.\n";
        print "<HR>";
        print "<A HREF=\"http://heartkorea.com/\"><I>Guest
Book</I></A>";
        exit(1);
}
sub filelock {                  # parameter is filename for lock
        my ($lockdir) = @_;
        mkdir ($lockdir, 0777) && return 'OK';
        -M $lockdir > 3/(24*60) && do {
                rmdir ($lockdir) || return 'FAIL';
        };
        for (1 .. 10) {
                mkdir ($lockdir, 0777) && return 'OK';
                sleep (1);
        }
        return 'BUSY';
}
sub fileunlock {
        my ($lockdir) = @_;
        -d $lockdir || return 'FAIL';
        rmdir ($lockdir) && return 'OK';
        return 'FAIL';
}
```

```
#### end of program ########################################
```

D.3 JEUS Web Server – JEUS 연동 환경설정 예

다음은 WEBMain.xml 과 `ws_engine.m` 에 대한 설정의 예이다

D.3.1 Sample WEBMain.xml file

<<WEBMain.xml>>

```
<?xml version="1.0"?>
<web-container  xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
     <context-group>
         <group-name>MyGroup</group-name>
         <group-docbase>webapps</group-docbase>
         <shared-session>true</shared-session>
         <session-timeout>20</session-timeout>
         <webserver-connection>
             <webtob-listener>
                 <listener-id>JEUS Web Server</listener-id>
                 <port>9900</port>
                 <webtob-address>localhost</webtob-address>
                 <webtob-home>
                     c:/jeus/webserver/docs</webtob-home>
                 <thread-pool>
                     <min>25</min>
                     <max>30</max>
                 </thread-pool>
             </webtob-listener>
         </webserver-connection>
     </context-group>
</web-container>
```

D.3.2 Sample sample.m file

<<sample.m>>

```
*DOMAIN
 JEUS_Web_Server

 *NODE
 webmain  WEBTOBDIR="C:/jeus/webserver",
```

```
          SHMKEY = 54000,
          DOCROOT="C:/jeus/webserver/docs",
          PORT = "8080",
          JSVPORT = 9900,
          HTH = 1

*SVRGROUP
htmlg   NODENAME = webmain, SvrType = HTML
jsvg    NODENAME = webmain, SVRTYPE = JSV

*SERVER
html     SVGNAME = htmlg, MinProc = 1, MaxProc = 2
MyGroup  SVGNAME = jsvg, MinProc = 25, MaxProc = 30

*URI
uri2     Uri = "/examples/",Svrtype = JSV

*EXT
htm     MimeType = "text/html", SvrType = HTML
```

E JEUS Web Server 환경설정 레퍼런스

E.1 소개

이 부록에서는 JEUS Web Server 환경설정 파일에서 사용되는 모든 항목에 대해서 상세하게 설명이 되어질 것이다.

아래의 각 절에 대해서 설명이 이루어질 것이다.

2. DOMAIN
3. NODE
4. VIRTUALHOST
5. SVRGROUP
6. SERVER
7. SERVICE
8. DIRECTORY
9. URI
10. ALIAS
11. DIRINDEX
12. LOGGING
13. AUTHENT
14. SSL
15. EXT
16. ACCESS

17. EXPIRES

각각의 항목은 4 개의 파트(개요, 필수항목, 선택항목, 사용예)로 나누어서 설명이 되어질 것이다.

E.2 DOMAIN 절

E.2.1 개요

DOMAIN 절은 독립적인 JEUS Web Server 시스템의 전반적인 환경에 대해 정의하는 부분이다. DOMAIN 의 이름은 호스트 이름과 함께 암호화 되어 JEUS Web Server 의 License 확인에 사용된다.

이제부터 DOMAIN 절의 항목들에 대해 자세히 살펴보겠다.

DOMAIN 절의 기본 환경설정 형식은 다음과 같이 정의한다:

```
Domain name       [,NliveInq]
```

E.2.2 필수항목

Domain name = <string>

string 형식으로 정의한다. 31 자를 초과하는 이름을 사용할 경우에는 컴파일 에러는 나지 않으나 내부적으로 31 자까지만 인식하게 된다. 31 자 이내로 이름을 설정하게 되어있는 기타 다른 절들의 항목에서도 이 점은 동일하다. Domain 이름은 호스트 이름과 함께 암호화 되어 시스템의 License 확인에 사용된다.

E.2.3 선택항목

NLiveInq = <numeric> (Default 30 Second)

JEUS Web Server 가 여러 Node 로 구성되었을 경우 Node 간 감시 Interval(초 단위) 시간을 정하는 항목이다. NLiveInq 시간이 짧으면 양 Node 간에 빈번한 감시로 시스템의 이상을 빨리 발견할 수 있으나 많은 통신량으로 인해 통신 부하가 발생할 수 있다. 반대로 NLiveInq 시간이 너무 길면 통신량은 적으나 Node 의 이상 상태를 즉시 감지 할 수 없다. 따라서 시스템 성능을 최적화 하기 위하여 네트워크 부하량, 업무의 중요도 등을 고려하여야 한다.

E.2.4 DOMAIN 절의 사용예

```
*DOMAIN
```

```
JEUS_Web_Server1        NLiveInq = 30
```

E.3 NODE 절

E.3.1 개요

이 절에서는 JEUS Web Server 를 이루는 각 Node 들에 대한 구체적인 환경설정이 이루어 진다.

NODE 절에는 다음과 같은 내용들이 정의될 수 있다:

- JEUS Web Server 시스템의 홈 디렉토리
- Service 할 HTML 문서들이 들어있는 디렉토리 트리의 최상위 디렉토리.
- 실행할 JEUS Web Server Process (HTH)의 개수
- 공유 메모리의 Key 값
- Port 번호 설정
- Listen IP Address 설정
- 기타 Logging 등.

JEUS Web Server 를 기동 시키면 Node 단위 별로 WSM, HTL, HTH Process 와 HTML, CGI 등의 실제 Service 를 수행하는 Server Process 들이 기동 된다. 이러한 Process 들을 기동하기 위해서는 실행 파일들이 존재하는 위치를 알아야 한다. 또한 JEUS Web Server Process 간 통신에 필요한 Directory 와 각종 Error 와 경고 메시지를 저장하는 Directory 도 지정할 수 있다. 그리고 각 항목별로 지시자들의 형태가 기록되어 있다. 이는 보통 Literal, Numeric, String 등의 형태로 나타나는데, 이들의 형태에 맞게 각 항목을 설정하여야 한다. Literal 은 보통 하나의 글자, Numeric 은 숫자, String 은 특정 단어를 말한다.

NODE 절의 기본 환경설정 형식은 다음과 같이 정의한다:

```
Node name          WEBTOBDIR=”JEUS Web Server-home-path”,
                   SHMKEY =shared-memory-segment-key,
                   DOCROOT =”JEUS Web Server-html-root”
                   [,User][,Group][,Admin][,HostName][,Method]
```

```
[,HTH][,HthQTimeout][,NodePort]
[Port][,SslFlag][,JSVPort][,RacPort]
[,SslName][,TimeOut][,CliChkIntval]
[,CacheSize][,CacheEntry] [,CacheRefreshHtml]
[,CacheRefreshDir][,KeppAlive]
[,KeepAliveTimeout][,KeepAliveMax][,AppDir]
[,PathDir][,TxLogDir][,SysLogDir][,UsrLogDir]
[,IconDir][,UserDir][,Errorlog][,EnvFile]
[,IndexName][,Listen][,DirIndex][,Options]
[,LanguagePrio][,Logging][,NodeType][,TcpGW]
[,NodeName][,MaxUser][,IPCPerm][,ListenBacklog]
[,SendBufferSize][,LimitRequestBody]
[,LimitRequestFields][,LimitRequestFieldSize]
[,LimitRequestLine][,ServerTokens]
[,IPCBasePort][,ErrorDocument][,DefaultCharset]
[,DefaultMinetype][,Expires][,ServiceOrder]
```

E.3.2 필수항목

Node name = <string>

string 으로 정의한다. 31 자를 초과하는 이름을 사용할 경우에는 컴파일 에러는 나지 않으나 내부적으로 31 자까지만 인식한다. 먼저 Node 의 물리적인 이름으로 Node 이름을 설정한다. 즉, 실제 등록된 Host 의 이름을 말한다. 예를 들어 UNIX 의 경우 “uname –n” 명령으로 각 Host 의 이름을 확인할 수 있다. 또한 해당 Node 명은 반드시 “/etc/hosts” 파일에 등록되어 있어야 한다. 하나의 Domain 은 하나 이상의 Node 로 이루어지므로, NODE 절에는 최소한 하나 이상의 Node 이름이 정의되어야 한다.

WEBTOBDIR = <literal>

Size : 255 이하

JEUS Web Server 가 설치되어 있는 홈 Directory 의 절대 경로명이다. 경로명은 환경 변수 WEBTOBDIR 과 동일한 값이 정의 되어야 한다. JEUS Web Server 관련 작업은 JEUS Web Server 디렉토리 하에서 모두 이루어진다

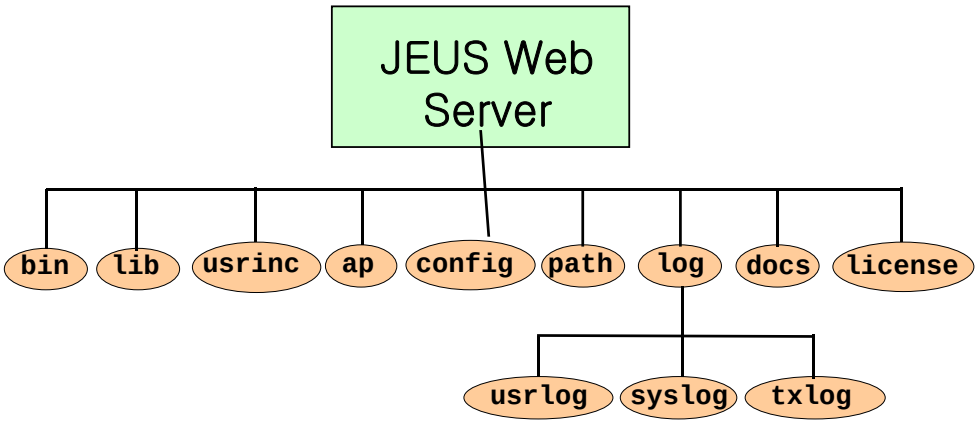

그림 9. JEUS Web Server 디렉토리 구조

bin/ : 실행 파일들이 위치

(wsm, wscfl, wsracd, wsgst, wsboot, wsdown, etc.)

lib/ : 라이브러리 파일이 위치

usrinc/ : API 함수들의 header 파일이 위치

ap/ : 사용자 프로그램이 위치

config/ : JEUS Web Server 환경 파일이 위치

path/ : Process 내부 통신을 위한 네임드 파이프(Named Pipe)가 생성

log/ : Log 파일들이 위치

docs/ : JEUS Web Server 에서 기본적으로 등록되는 html 문서 위치

license/ : JEUS Web Server License 파일이 위치

이러한 bin, config, lib, docs, license, log, path 등의 하부 디렉토리 구조를 가지는 상위 Directory 명을 WEBTOBDIR 에 정의한다.

SHMKEY = <numeric> (32768 ~ 262143)

공유 메모리 세그먼트(Shared Memory Segment)를 가리키는 값이다. JEUS Web Server 를 이루는 Process 들이 서로 정보를 공유하기 위해 공유 메모리 Key 값을 정의할 필요가 있다. 공유 메모리 Key 값을 정의하기 전에 이 Key 값들이 다른 프로그램 또는 다른 업무에서 사용되는지 반드시 확인해야 한다. 그렇지 않으면 JEUS Web Server Booting 시에 이 프로그램과 충돌을 일으켜서 실행이 되지 않는다

현재 JEUS Web Server 에서 정의되는 Shared Memory 의 Key 값은 최소 32768 에서 최대 262143 까지이다. 따라서 이 범위 내에 있는 값을 이용하여야 한다.

DOCROOT = <literal>

JEUS Web Server 가 웹을 통해 Service 하는 모든 문서를 포함하는 루트 Directory 의 경로명이다. JEUS Web Server 는 DOCROOT 가 지정한 Directory 를 최상위로 하여 문서를 Service 하게 된다. Client 가 요구한 URL 은 DOCROOT 의 경로 뒤에 추가되어 실제 경로명을 이루게 된다. JEUS Web Server 는 이 경로명을 가지고 파일에 접근한다.

E.3.3 선택항목

HTH = <numeric> (Default Number: 1, Max Number: 20)

JEUS Web Server 에서 가장 중요한 역할을 담당하고 있는 HTH (HTTP Request Handler) Process 의 개수를 설정한다. HTH 는 실제로 Client Browser 와 JEUS Web Server 내부 Service Process 사이를 중계하는 Process 이다. 즉, Client 의 요청을 받아 Service 를 받을 수 있도록 적당한 Process 에 넘겨주고 다시 처리된 결과를 수신하여 Client 에게 되돌려준다. JEUS Web Server 에서는 모든 Client 가 이 HTH Process 에 연결되도록 설계되어 있다. 따라서 많은 수의 동시 접속자를 유지하기 위해서는 이 수를 적당히 늘려주는 것도 필요하다. 그러나 보통 하나의 HTH Process 가 적어도 800 개 이상의 Client 를 수용할 수 있기 때문에 하나의 HTH 로도 큰 문제는 되지 않을 것이다. 만약 1000 명 이상의 동시 Client 를 수용하여야 하는 System 에서는 두개 이상의 HTH Process 를 구동하여야 한다. 현재 JEUS Web Server 에서는 최대 20 개까지의 HTH Process 를 구동하는 것이 가능하다. 따라서 약 16000 명의 사용자를 동시에 처리하는 것이 가능하다.

PORT = <literal> (Default Port: 80)

JEUS Web Server 가 Listen 하는 Port 번호를 설정한다. 이는 기본적으로 Web Service 를 하기 위해서는 반드시 필요한 설정으로, 이 항목은 반드시 존재하여야 한다. 보통 일반적인 Web Server 는 80 Port 를 이용한다. 그러므로 이 항

목을 설정하지 않았을 경우에는 default 로 80 Port 가 설정된다. 그러나 다른 용도나 Internet 등의 경우에는 임의의 Port 를 설정하여 이용할 수 있다. JEUS Web Server 설치 CD-ROM 에서 제공하는 예제 환경파일에도 root 권한으로 설치를 할 경우에는 PORT 항목에 80 으로 setting 이 되지만 일반 user 권한으로 설치할 경우에는 Permission 문제로 인하여 PORT 설정이 정상적으로 안될 경우가 많으므로 주의하여 반드시 수동으로 원하는 Port 번호를 지정하기 바란다.

Virtual Host 를 사용하는 경우, Port 번호는 VHOST 절에서 재정의 해야 한다. 또한, JEUS Web Server 의 경우 여러 개의 Port 를 동시에 정의하는 것도 가능하다. 현재는 최대 10 개 까지의 Port 를 동시에 설정하는 것이 가능하다. 즉, PORT 의 설정을 통해서 여러 개의 Port 가 동시에 Listen 하는 것이 가능하다는 것이다. 이에 대한 간단한 예는 다음과 같다.

```
PORT = “9090,9091”
```

위와 같은 Setting 이 이루어지면 JEUS Web Server 는 9090 과 9091 Port 에서 모두 Web Service 를 받아들일 수 있다.

참고: 한가지 주의할 점은 이것은 Listen 항목과 동시에 운영할 수 없다는 것이다. Listen 항목에서 특정 Port 를 지정하는 경우 이 Port 만을 이용하게 된다. 즉, 중요한 것은 Listen 항목에서 지정된 Port 가 PORT 항목에서 지정된 것보다 우선된다는 것이다. 물론 Listen 항목에서도 여러 개의 Port 를 지정하는 것이 가능하다. 이에 대한 예는 Listen 항목의 설명에서 할 것이다.

E.3.4 User 와 Group 설정

JEUS Web Server 에서는 시스템의 보안을 위하여 JEUS Web Server 의 실제 실행 Process 에 대한 권한 설정을 할 수 있게 하였다. 만약 Web Server 의 실행 Process 들이 root 권한으로 실행되는 경우, 이에 문제가 발생하게 되면 시스템 전체에 큰 피해를 주게 될 수도 있다. 따라서, JEUS Web Server 에서는 이를 방지하고자 JEUS Web Server 의 실행 Process 들을 root 권한이 아닌 다른 권한으로 실행할 수 있게 하였다.

그러나 이러한 실행 권한의 변환은 root 만이 가능하기 때문에 이를 설정하기 위해서는 반드시 root 로 작업을 하여야 한다. 또한, 자신이 설정하고자 하는 user 와 group 이 시스템에 설정되어 있어야 한다. 이에 대한 사항은 자신이 운영하는 운영체제의 Manual 을 참조하기 바란다.

User = <string>

설정된 User 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Group = <string>

설정된 Group 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 Group 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Admin = <string>

관리자의 정보를 나타낸다. 여기에는 관리자에게 연락할 수 있는 E-mail 주소를 설정할 수 있다.

HostName = <literal>

이 항목을 설정하면 Http Response Header 의 host name Field 에 기록을 남겨준다. JEUS Web Server 가 설치된 machine 의 Domain name 을 "www.tmax.co.kr"과 같이 넣어주면 된다.

NodeName = <literal>

이 항목의 값으로는 해당 머신의 HostName 을 적어 주면 되는데, 특별히 $(NODENAME)이라고 적어주면, Web Server 가 자동적으로 해당 머신의 HostName 을 적용하게 된다. 이 NodeName 항목이 추가됨으로 해서 Virtual Node 개념이 추가 되었으며, 그동안 노드명의 제약사항이였던 31 자 이상의 이름이나, 한글 또는 공백문자가 포함된 HostName 도 사용할 수 있게 되었다.

NodePort = <numeric> (Default Port: 7777)

Node 간 연결 Port 번호를 설정한다. Node 간 통신은 이 Port 번호를 통하여 이루어 진다. JEUS Web Server 를 여러 Node 에 설치하였을 때, 각 Node 에 있는 JEUS Web Server 는 서로 연결을 맺고 정보를 공유한다. 따라서 이 Port 는 Multi Node 에 JEUS Web Server 를 설치한 경우 반드시 필요하다.

JSVPort = <numeric> (Default Port: 9999)

JEUS Web Server 와 Java Servlet 수행 Server 간의 연결 Port 번호이다.

RacPort = <numeric> (Default Port: 3333)

JEUS Web Server 가 여러 Node 로 구성되어 있을 경우, Node 관리 차원에서 Node 간 통신을 위한 Port 번호를 정의한다. 위의 NodePort 와는 달리 이것은 관리 Process 중 하나인 wsracd Daemon 에서 사용하는 Port 번호이다.

ClichkIntval = <numeric> (Default Number: 60 second)

JEUS Web Server 에서 접속한 Client 가 살았나 죽었나를 확인할 필요가 있을 때 설정하는 항목이다. 항목 시간 간격을 설정함으로써 JEUS Web Server 는 Client 의 생존 여부를 주어진 시간 간격마다 확인하고 죽었을 경우 JEUS Web Server 쪽에서 접속을 끊는다.

CacheSize = <numeric> (Default Size: 128 Kbyte)

JEUS Web Server 는 Server 내부 Caching 기능을 지원한다. 따라서 많은 Web Application 이 용이하게 이루어 질 수 있다. 예를 들어 세션 정보를 관리해야 하는 경우 기존의 방식처럼 쿠키를 쓰지 않고 JEUS Web Server 가 내부적으로 제공하고 있는 Cache 에 정보를 저장함으로써 더 편리하고 효율적인 작업을 수행할 수 있다. 이러한 내부 Cache 의 사이즈를 필요에 따라 조정할 수 있다. 여기서는 Cache 의 한 엔트리의 크기로서 기본단위는 Kbyte 이다.

CacheEntry = <numeric> (Default Number: 128)

Cache 의 총 Hashing Key 엔트리 개수를 설정한다. JEUS Web Server 에서는 Hashing 방식을 이용하기 때문에 이의 값에 따라서 Cache 기능의 성능이 영향을 받게 된다. 만약 이 값이 적게 설정되면 Hashing Key 값이 적게 되어 Key 는 쉽게 찾지만, 각 Key 에서의 값을 찾는 문제가 발생하고 Key 값이 많게 되면 다양한 Key 에 대한 값이 나오지만 각 Entry 가 적게 되어 쉽게 찾을 수 있다. JEUS Web Server 에의 기본 설정값은 128 이다. 이 값을 변경하여도 큰 영향은 없지만 가급적 128 개의 Hashing 값을 권장하는 바이다.

CacheRefreshHtml = <numeric> (Default Number: 0 second)

HTML file 에 대한 cache refresh time 을 설정한다.

CacheRefreshHtml = <numeric> (Default Number: 0 second)

DirIndex 에 대한 cache refresh time 을 설정한다.

KeepAlive = <literal>

KeepAlive 는 HTTP 1.1 스펙에 포함된 기술로, 어떤 사용자가 웹사이트에 접속할 때, 그들이 웹사이트내의 다른 웹페이지를 읽어 들이기 위해 곧 다시 접속을 시도하려는 경우가 매우 많다는 것을 쉽게 예상할 수 있을 것이다.

이럴 경우에 불필요한 시간 지연이 없도록 하려면 이 항목을 지정함으로써 접속을 단절하지 않고 유지할 수 있다.

KeepAliveMax = <numeric> (Default Number : 9999)

보통은 하나의 Client 가 한 개 이상의 요구를 연속적으로 Server 에 요청하는 경우가 많다. 이러한 경우 매 요구마다 연결을 다시 맺어야 한다면 비효율적일 것이다. 따라서 일정 개수의 요구는 처음 커넥션을 유지한 상태로 Service 를 하고 커넥션을 끊도록 한다. 커넥션을 끊기 전에 들어주는 요구의 개수를 KeepAliveMax 에서 지정한다.

KeepAliveTimeout = <numeric> (Default Number : 60)

하나의 Client 가 불필요하게 커넥션을 오래 잡고 있는 경우를 막기 위해서 요구간 시간 간격이 일정 시간 이상이 되면 커넥션을 끊을 수 있도록 설정할 수 있다. 이러한 요구간 시간 간격은 KeepAliveTimeout 값으로 설정한다. Default 로 설정되는 값은 60 이며 단위는 second 이다.

Timeout = <numeric> (Default Number : 300)

사용자가 접속을 하여 Data 를 내려 받거나, 사용자의 요구가 내려 받는 시간을 지정하는 것이 가능하다. 이 때 이 Timeout Field 를 이용하게 되는데, 이 Field 를 통해서 사용자의 최대 접속시간을 지정할 수 있다. 이는 사용자와 맺은 연결이 문제가 생겨 계속해서 의미 없는 Data 전송이 발생할 때 이를 방지하기 위해서 이용한다. Default 로 설정되는 값은 300 이며 단위는 second 이다.

UserDir = <literal>

JEUS Web Server 를 통해 여러 사용자를 동시에 서비스 하려는 경우 필요하다. 이 때 들어가는 값은 각 사용자의 Directory 의 이름이다. 이를 설정하면 JEUS Web Server 는 각 사용자의 directory 를 찾아서 서비스를 시작한다.

AppDir = <literal>

JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한

다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

E.3.5 기본 Log Directory 의 설정

JEUS Web Server 에서는 기본적으로 Log 정보를 남기기 위하여 반드시 설정하여야 하는 directory 들이 존재한다. 이는 JEUS Web Server 에 장애가 발생하거나 혹은 JEUS Web Server 로 특정 Service 를 하였을 때 문제가 생기는 경우, 이를 처리하기 위하여 이에 대한 기록을 남기는 것이다. 이는 환경파일에서 따로 설정하지 않아도 기본적으로 WEBTOBDIR 로 설정된 Directory 밑에 log directory 에 만들어지게 되는데, 주의할 점은 JEUS Web Server 의 Booting 전에 반드시 이에 대한 Directory 가 실제로 존재하는지를 확인하여야 한다는 것이다. 만약 이 Directory 가 없으면 JEUS Web Server booting 시에 이를 찾다가 파일을 찾지 못한다는 에러를 리턴하면서 booting 이 되지 않는다. 그러므로 어떤 오류로 인하여 log Directory 가 만들어지지 않았다면 반드시 Log Directory 를 먼저 만들어 주고 JEUS Web Server 를 실행하여야 한다. 반드시 설정하여야 하는 Directory 에는 트랜젝션 관련 로그를 기록하는 디렉토리(default : (WEBTOBDIR)/log/txlog)와 시스템 관련 로그를 기록하는 디렉토리(default : (WEBTOBDIR)/log/syslog)가 있는데 이들 Directory 들은 Install script 를 이용하여 JEUS Web Server 를 설치하였다면 default 경로로 자동으로 생성된다. 이 경로는 사용자가 다시 재정의하여 사용할 수 있는데 그 방법은 아래 설명하는 SysLogDir 이라는 항목을 설정하면 된다.

SysLogDir = <literal> (Default Path : (WEBTOBDIR)/log/syslog)

시스템 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다. 시스템 메시지란 wsm, htl, hth, htl, hth 등 JEUS Web Server 기동의 핵심 Process 들이 발생한 메시지들과 시스템 내부적으로 발생한 메시지들을 일컫는다. 이 항목은 오직 SysLogDir 를 재정의하여 사용하기 위한 것으로, 이 항목을 설정하지 않는 경우에는 Default Path 에 Log 가 남는다. Default Path 를 수정하여 사용할 경우 주의를 요한다.

UsrLogDir = <literal> (WEBTOBDIR)/log/usrlog

사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

IndexName = <literal>

Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때 기본적으로 Service 되는 파일 이름을 설정한다. 따로 설정하지 않으면 `index.html` 이 설정된다.

Options = <literal>

Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때의 동작을 지정한다. 보통 사용자가 특정 URI 를 보내고 디렉토리 이름만 요구 하였다면 사용자에게 Directory 의 내용을 보여주는 것이 가능하다. 물론 원하지 않는 다면 보여 주지 않을 수 있다. 기본적인 설정은 보여주지 않는 것으로 되어 있다. 이 때 '+' 나 '−' Option 들을 이용하는데, 만약 디렉토리 정보를 보여주길 원한다면 "+Index"와 같이 설정하면 된다.

이외에도 그 Node 에서 CGI 를 쓰고 싶지 않은 경우 −CGI 와 같이 설정하면 CGI 가 수행되지 않게 된다. 즉, + 나 − Option 을 통해서 많은 Service 들 혹은 기능들의 수행여부를 결정할 수 있다는 것이다. 이에 이 부분에서 설정 가능한 Service 들과 기능들은 다음과 같다. 만약 ALL 을 기입하면 모든 기능에 대한 설정을 하는 것이다. 즉, -ALL 은 모든 기능을 이용하지 않겠다는 것이고 , +ALL 은 모든 기능을 이용하겠다는 것이다.

Service: HTML, CGI, SSI, PHP, JSV

Function: INDEX, INCLUDE

Method = <literal>

Client 가 보내는 Request Method 에 대한 정의를 할 수 있다. 이 때 기본적으로 GET, POST, HEAD 등이 있어 이를 기본적으로 지원하고 만약 이들 중 특정 Method 를 쓰고 싶지 않은 경우 − Option 을 이용하여 제거할 수 있다.

Listen = <literal>

JEUS Web Server 가 Booting 될 때, 원하는 IP Address 에서 연결을 맺도록 할 수 있다. 즉, 여러 개의 IP Address 를 가진 Server 에서 자신이 원하는 IP Address 에서만 Service 를 원하는 경우, 이 값을 정할 수 있다. 이 때 여러 개의 것을 복수로 설정하는 것도 가능하다.

DirIndex = <literal>

뒤의 DIRINDEX 절에서 설정하는 디렉토리 인덱스의 이름을 적어준다.

LanguagePrio = <literal>

접속 Client 가 사용 언어를 지정하지 않았을 경우 Server 쪽에서 지정된 언어 순서대로 Multiview request 등의 처리가 이루어지도록 한다.

Logging = <literal>

뒤의 Logging 절에서 설정하는 Logging Name 을 써준다. 이 이름을 가지고 이 Node 에서 그에 해당하는 Log 를 남기게 되는 것이다.

ErrorLog = <literal>

오류 발생시 설정하는 Logging 정보 이름을 써준다. 이 이름 또한 뒤의 Logging 절에서 Logging Name 으로 사용된다.

EnvFile = <string>

JEUS Web Server 에서 특정 정보를 읽어 들일 필요가 있는 경우 이용된다. 즉, 어떤 파일에 변수와 변수에 대한 정보를 기록한 후, JEUS Web Server 기동 시에 이 정보를 읽어야 하는 경우 이 EnvFile 에 등록된 파일을 읽어 들인다. 즉, JEUS Web Server 가 기동 시에 이 파일을 읽어서 그 정보를 가지고 있게 된다.

SSLFLAG = string (Y | N) (default : N)

JEUS Web Server 에서 SSL 을 이용할 때 반드시 지정하여야 하는 항목이다. 이 SSLFLAG 가 Y 상태이면, 그 Node 에서 SSL 을 이용하겠다는 것이고, N 상태이면, 이용하지 않는다는 것이다. 기본 설정은 SSL 을 이용하지 않는 것으로 되어 있다.

SSLNAME = <string>

JEUS Web Server 에서 SSL 을 이용하는 경우, 이에 대한 설정을 나타내는 것이다. 이는 반드시 SSLFLAG 가 on 이 된 상태에서 적용되어야 하며 off 나 설정이 되지 않은 상황에서는 아무런 의미가 없다. 이 때 지정되는 이름은 반드시 SSL 절에서 선언이 되어 있어야 한다.

MaxUser = <numeric>

서버 프로세스에 속한 노드의 최대 동시 접속자 수를 설정한다.

IPCPerm = <numeric> (Default Number : 0700)

IPCPERM(inter-process communication permission mask)은 Web Server 시스템에 대한 관리자가 아닌 개발자와 같은 다른 사용자가 wsdown 이나 특정

프로세스를 기동 및 종료를 할 수 있으며 이를 wsadmin 를 통해 확인할 수 있도록 한다. Unix 시스템 환경 하에서는 관리자 개인이나 그룹,기타에게 각각 파일 접속 제어(판독기능/기록기능/수행기능)를 지정 할 수 있다. 즉, default 인 경우에는 다른 사용자들은 위의 기능을 사용할 수 없으나 IPCPERM 이 0777 인 경우에는 다른 사용자들은 위의 모든 기능을 사용할 수 있다.

ListenBacklog = <numeric> (Default Number : 511)

접속을 기다리는 큐(queue)의 길이를 제한하는 것으로, 보통은 거의 필요하지 않지만 서버가 대량의 접속 시도를 한꺼번에 날려주는 TCP SYN 해킹을 당하고 있다면 유용하게 사용 될수 있을 것이다. 만일, Request 가 WebtoB 가 처리할수 있는 것보다 빠르게 요구되어지면, Listen queue 는 Overflow 가 되는데, 이때 추가적인 Request 는 Listen queue 에 여유가 생길때까지 OS 에 의해 거절되어진다.

SendBufferSize = <numeric> (Default : OS default)

TCP 전송 Buffer 의 크기를 설정하는 것으로, 이 항목을 이용하면 특정한 환경에서 동작 속도를 향상시킬 수 있다. 0 의 값은 OS default 값을 사용함을 의미한다.

LimitRequestBody = <numeric> (Default Number: 0 bytes)

클라이언트의 요청시 HTTP 프로토콜을 통해 서버가 제공할 수 있는 Request Body 크기를 바이트 단위로 정의하는 것으로, 0 의 값은 크기에 제한이 없음을 의미한다.

LimitRequestFields = <numeric> (Default Number : 100)

클라이언트의 요청시 허용되는 HTTP Request header field 의 수를 설정한다. 0 의 값은 제한이 없음을 의미한다.

LimitRequestFieldSize = <numeric> (Default Numner : 8190 bytes)

클라이언트의 요청시 허용되는 각 HTTP Request header field 의 크기를 설정한다. 최대 허용되는 값은 8190 이다.

LimitRequestLine = number> (Default Numner : 8190 bytes)

클라이언트의 요청시 허용되는 HTTP Request line 의 최대 크기를 설정한다. 최대 허용되는 값은 8190 이다.

ServerTokens = <literal> (Default : "Minimal")

HTTP 응답 헤더의 Server 에 관한 정보를 어떻게 다룰지 결정한다.

"Off"	server에 관한 정보를 보내지않음.
"Prod[uctOnly]"	ex. WebtoB
"Min[imal]"	ex. WebtoB/3.1.5
"OS"	ex. WebtoB/3.1.5 (LINUX_i386)
"Full"	ex. WebtoB/3.1.5 (LINUX_i386)
"Custom=xxx/x.x"	ex. xxx/x.x

ServiceOrder = <literal> (Default : "uri,ext")

HTTP 요청으로부터 해당 Server 와 Service 를 결정할때, URI 절과 EXT 절의 우선순위를 결정한다. Vhost 절에 이 항목이 설정되지 않은 경우는 Node 절에 설정된 값이나 기본값을 Vhost 가 따르게 된다.

"uri,ext" : URI, EXT Section 순으로 Server 와 Service 를 결정한다.

"ext,uri" : EXT, URI Section 순으로 Server 와 Service 를 결정한다.

DefaultCharset = <literal> (Default : none)

HTTP header 중에서 Content-Type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 절에서 적용되는 우선 순위는 Node < Vhost < SvrGroup < Directory 순이다.

"On"	기본 character set인 ISO-8859-1로 설정한다.
"Off"	해당 기능을 중단한다.
"_charset_"	사용자가 기술한 _charset_으로 설정한다.

DefaultMimetype = <literal> (Default : "text/html")

MIME-Type 을 결정할 수 없는 문서의 Default Content-Type 을 설정한다. Default Content-Type 은 SvrGroup, Vhost, Node 절의 순으로 결정된다.

IPCBasePort = <numeric> (Default Port : 6666)

Web Server 에서 내부 프로세스간 IPC 통신을 하기 위해서 기본적으로 특정 포트(6666)를 사용하는데, IPCBasePort 항목을 통해 해당 포트를 변경할 수

있다. 현재 Windows 에서만 지원되며, 이전 버전에서는 환경변수에 WEBTOB_WINDOWS_PORT 를 설정함으로써 같은 기능을 제공하였다. 두 가지가 같이 설정이 된 경우에는 환경변수에 설정된 Port 가 우선순위를 갖는다.

Expires = <string>

Expires 이름을 설정한다.

E.3.6 NODE 절의 사용예

```
*NODE
webmain     WEBTOBDIR = "/usr/local/webserver",
            SHMKEY = 69000,
            DOCROOT = "/usr/local/webserver/docs",
            APPDIR = "/usr/local/webserver/ap",
            HOSTNAME = "webmain.tmax.co.kr",
            PORT = "8080",
            Listen = "192.168.63.2:8080,192.1.1.9:8080",
            User = "nobody",
            Group = "nobody",
            Hth = 1,
            HthQTimeout = 10,
            JSVPORT = 9099,
            NodePort = 8888,
            RacPort = 3333,
            TimeOut = 100,
            CliChkIntval = 60,
            CacheSize = 128,
            CacheEntry = 128,
            KeepAliveTimeOut = 30,
            KeepAliveMax = 100,
            IndexName = "Index.html",
            Options = "-Index",
            LOGGING = "log1",
            ERRORLOG = "log2",
            PathDir = “/usr/local/webserver/path",
             SysLogDir = “/usr/syslog/webserver/log/syslog",
            UsrLogDir = “/usr/usrlog/webserver/log/usrlog",
            IconDir = “/usr/icons/webserver/icons",
            UserDir = “public_html",
```

```
            EnvFile = webserver_env,
            DirIndex = "Index",
            Method = "GET, POST, HEAD",
            LanguagePrio = "kr"
            IPCBasePort = 7777
```

E.4 VIRTUALHOST 절

E.4.1 개요

이 절에서는 JEUS Web Server 로 Virtual Hosting 이 필요한 경우, 이에 대한 환경 설정을 한다. Virtual Host 기능은 실제로는 하나의 JEUS Web Server 가 동작하지만 각기 다른 URL 로 다른 문서를 제공하도록 함으로서 마치 여러 개의 Server 가 Service 를 제공하는 것처럼 보이도록 하는 기능이다.

VHOST 절에는 다음과 같은 내용이 정의된다:

- VirtualHost 에 해당하는 Node 이름
- VirtualHost 에 해당하는 호스트 이름
- 각 Virtual Host 가 Service 할 HTML 문서들이 들어있는 최상위 디렉토리

또한 NODE 절에서 정의한 다음 내용은 다시 새롭게 정의할 수 있다.

- User, Group, Admin 에 대한 정의 항목
- 각종 로그 메시지 디렉토리 경로명
- 디렉토리 아이콘 경로명과 Indexing 방식

앞의 NODE 절에서 이미 정의된 값들도 VHOST 절에서 다시 정의된 값으로 수정 된다.

VHOST 절의 기본 환경설정 형식은 다음과 같이 정의한다:

```
VHost name        NODENAME,
                  HOSTNAME,
                  DOCROOT
                  [,User][,Group][,Admin][,Method][,AppDir]
                  [,Listen][,Port][,SSLFLAG][,SSLNAME]
```

```
[,UsrLogDir][,IconDir][,UserDir]
[,EnvFile][,IndexName][,DirIndex]
[,Options][,PathDir][,AuthentName]
[,Logging][,HostAlias][,ErrorDocument]
[,Errorlog][,NodeType][,DefaultCharset]
[,DefaultMimetype][,Expires][,ServiceOrder]
[,KeepAlive][,KeepAliveTimeout]
[,KeepAliveMax][,Timeout]
```

E.4.2 필수항목

VHOST name = <string>(31 자 이내)

string 으로 사용자 임의로 정의할 수 있다.

NODENAME = <string>

Virtual Host 가 속해있는 Node 의 이름을 적어준다. 이 Node 이름은 NODE 절에 정의되어 있어야 한다.

HOSTNAME = <string>

Virtual Host 에 접근할 때 유저가 사용할 호스트 이름을 적어준다. Name-based Virtual Hosting 을 하는 경우 각각의 VHOST 절에서 호스트 이름을 다르게 정의한다. 이 경우는 호스트 이름으로 Virtual Host 가 구분된다.

DOCROOT = <string>

정의된 Virtual Host 가 Service 하게 될 HTML 문서가 있는 최상위 디렉토리 경로명을 적어준다.

E.4.3 선택항목

VHOST 절의 선택 항목은 앞의 NODE 절의 선택 항목 중 일부분이며 그 내용은 동일하다. 단지 설정된 virtual host 에 대해 재정의 된다. VHOST 절에서 정의할 수 있는 항목들은 다음과 같다:

PORT = <literal> (Default Port: 80)

JEUS Web Server 가 Listen 하는 Port 번호를 설정한다. 이는 기본적으로 Web Service 를 하기 위해 반드시 필요한 설정으로, 이 항목은 반드시 존재하여야 한다. 보통 일반적인 Web Server 는 80 Port 를 이용한다. 그러나 다른 용도나 Internet 등의 경우에는 임의의 Port 를 설정하여 이용할 수 있다. JEUS Web

Server 의 경우 여러 개의 Port 를 동시에 정의 하는 것도 가능하다. 현재는 최대 5 개 까지의 Port 를 동시에 설정하는 것이 가능하다. 이 절의 기능은 Node 의 것과 같다.

즉, PORT 의 설정을 통해서 여러 개의 Port 가 동시에 Listen 하는 것이 가능하다는 것이다. 이에 대한 간단한 예는 다음과 같다:

```
PORT = “9090,9091”
```

위와 같은 Setting 이 이루어지면, JEUS Web Server 는 9090 과 9091 Port 에서 모두 Web Service 를 받아들일 수 있다.

Listen = <literal>

JEUS Web Server 가 Booting 될 때, 원하는 IP Address 에서 연결을 맺도록 할 수 있다. 즉, 여러 개의 IP Address 를 가진 Server 에서 자신이 원하는 IP Address 에서만 Service 를 원하는 경우 이 값을 설정하여 정할 수 있다. 이 때 여러 개의 IP 을 복수로 설정하는 것도 가능하다. 이 절의 기능은 Node 의 것과 같다.

SSLFLAG = <string> (Y | N)

JEUS Web Server 에서 SSL 을 이용할 때 반드시 지정하여야 하는 항목이다. 이 SSLFLAG 가 Y 상태이면, 그 Node 에서 SSL 을 이용하겠다는 것이고, N 상태이면, 이용하지 않는다는 것이다. 기본 설정은 SSL 을 이용하지 않는 것으로 되어 있다.

SSLNAME = <string>

JEUS Web Server 에서 SSL 을 이용하는 경우, 이에 대한 설정을 나타내는 것이다. 이는 반드시 SSLFLAG 가 on 이 된 상태에서 적용되어야 하며 off 나 설정이 되지 않은 상황에서는 아무런 의미가 없다. 이 때 지정되는 이름은 반드시 SSL 절에서 선언이 되어 있어야 한다.

E.4.4 User 와 Group 설정

JEUS Web Server 에서는 시스템의 보안을 위하여 JEUS Web Server 의 실제 실행 Process 에 대한 권한 설정을 할 수 있게 하였다. 만약 Web Server 의 실행 Process 들이 root 권한으로 실행되는 경우, 문제가 발생하게 되면, 시스템 전체에 큰 피해를 주게 될 수도 있다. 따라서, JEUS Web Server 에서는 이를 방지하고자 JEUS Web Server 의 실행 Process 들을 root 권한이 아닌 다른 권한으로 실행할 수 있게 하였다.

그러나 이러한 실행 권한의 변환은 root 만이 가능하기 때문에 이를 설정하기 위해서는 반드시 root 로 작업을 하여야 한다. 또한, 자신이 설정하고자 하는 user 와 Group 이 시스템에 설정되어 있어야 한다. 이에 대한 사항은 자신이 운영하는 운영체제의 Manual 을 참조하기 바란다.

User = <string>

설정된 User 의 권한으로 JEUS Web Server 가 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Group = <string>

설정된 Group 의 권한으로 JEUS Web Server 가 요구를 수행하게 된다. Client 요구 실행을 위해 따로 Group 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Admin = <string>

관리자의 정보를 나타낸다. 여기에는 관리자에게 연락할 수 있는 E-mail 주소를 설정할 수 있다.

AppDir = <literal>

JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

IndexName = <string>

Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때 기본적으로 Service 가 되는 파일 이름을 설정한다. 따로 설정하지 않으면 `index.html` 이 설정된다.

DirIndex = <literal>

뒤의 DIRINDEX 절에서 설정하는 디렉토리 인덱스의 이름을 적어준다.

Method, UsrLogDir, EnvFile, ErrorDocument, Logging Option 등의 것들은 NODE 절에서 정의된 것과 같은 역할을 한다.

ServiceOrder = <literal> (Default : "uri,ext")

HTTP 요청으로부터 해당 Server 와 Service 를 결정할때, URI 절과 EXT 절의 우선순위를 결정한다. Vhost 절에 이 항목이 설정되지 않은 경우는 Node 절에 설정된 값이나 기본값을 Vhost 가 따르게 된다.

"uri,ext" : URI, EXT Section 순으로 Server 와 Service 를 결정한다.

"ext,uri" : EXT, URI Section 순으로 Server 와 Service 를 결정한다.

DefaultCharset = <literal> (Default : none)

HTTP header 중에서 Content-Type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 절에서 적용되는 우선 순위는 Node < Vhost < SvrGroup < Directory 순이다.

"On"	기본 character set인 ISO-8859-1로 설정한다.
"Off"	해당 기능을 중단한다.
"_charset_"	사용자가 기술한 _charset_으로 설정한다.

DefaultMimetype = <literal> (Default : "text/html")

MIME-Type 을 결정할 수 없는 문서의 Default Content-Type 을 설정한다. Default Content-Type 은 SvrGroup, Vhost, Node 절의 순으로 결정된다.

KeepAlive = <literal>

KeepAlive 는 HTTP 1.1 스펙에 포함된 기술로, 어떤 사용자가 웹사이트에 접속할 때, 그들이 웹사이트내의 다른 웹페이지를 읽어 들이기 위해 곧 다시 접속을 시도하려는 경우가 매우 많다는 것을 쉽게 예상할 수 있을 것이다.

이럴 경우에 불필요한 시간 지연이 없도록 하려면 이 항목을 지정함으로써 접속을 단절하지 않고 유지할 수 있다.

KeepAliveMax = <numeric> (Default Number : 9999)

보통은 하나의 Client 가 한 개 이상의 요구를 연속적으로 Server 에 요청하는 경우가 많다. 이러한 경우 매 요구마다 연결을 다시 맺어야 한다면 비효율적일 것이다. 따라서 일정 개수의 요구는 처음 커넥션을 유지한 상태로 Service 를 하고 커넥션을 끊도록 한다. 커넥션을 끊기 전에 들어주는 요구의 개수를 KeepAliveMax 에서 지정한다.

KeepAliveTimeout = <numeric> (Default Number : 60)

하나의 Client 가 불필요하게 커넥션을 오래 잡고 있는 경우를 막기 위해서 요구간 시간 간격이 일정 시간 이상이 되면 커넥션을 끊을 수 있도록 설정할 수 있다. 이러한 요구간 시간 간격은 KeepAliveTimeout 값으로 설정한다. Default 로 설정되는 값은 60 이며 단위는 second 이다.

Timeout = <numeric> (Default Number : 300)

사용자가 접속을 하여 Data 를 내려 받거나, 사용자의 요구가 내려 받는 시간을 지정하는 것이 가능하다. 이 때 이 Timeout Field 를 이용하게 되는데, 이 Field 를 통해서 사용자의 최대 접속시간을 지정할 수 있다. 이는 사용자와 맺은 연결이 문제가 생겨 계속해서 의미 없는 Data 전송이 발생할 때 이를 방지하기 위해서 이용한다. Default 로 설정되는 값은 300 이며 단위는 second 이다.

E.4.5 VHOST 절의 사용예

```
*VHOST
vhost1      DOCROOT = "/usr/local/webserver/docs/vhost_docs",
            NODENAME = webmain,
            HOSTNAME = "test.tmax.co.kr",
            PORT="8080",
            User = "nobody",
            Group = "nobody",
            UsrLogDir = “/usr/usrlog/webserver/vhost/log/usrlog",
            IconDir = "/usr/icons/webserver/vhost/icons",
            UserDir = “/usr/VHost/webserver/vhost",
            EnvFile = webserver_env,
            IndexName = "test.html",
            LOGGING = "log1",
            ERRORLOG = "log2
```

E.5 SVRGROUP 절

E.5.1 개요

JEUS Web Server 를 통해 응용 Server Process 를 접근하는 경우 Server Process 의 논리적인 연관성에 따라 이들을 그룹으로 관리할 필요가 있게 된다. 이 절에서는 이러한 그룹에 대한 환경 설정이 이루어 진다. Node 이름, Server 의 종류, 호스트의 이름 등을 등록한다.

SVRGROUP 절에는 다음과 같은 내용이 정의된다:

- Server Group 이 속한 Node 의 이름.
- Server Group 이 제공하는 Service 의 타입

이 밖에 NODE 절이나 VHOST 절에서 정의한 내용이 Server Group 에 따라 새롭게 정의할 수 있으며 데이터베이스를 사용하는 경우 데이터 베이스 접근과 관련된 정보들이 정의될 수 있다.

SVRGROUP 절에는 기본적으로 다음과 같은 내용이 정의될 수 있다.

```
Server Group name       NODENAME,
                        SVRTYPE
                        [,VhostName][,Cousin][,Backup][,Load]
                        [,AppDir][,UsrLogDir][,DBName][,OpenInfo]
                        [,CloseInfo][,Logging][,EnvFile]
                        [,MinTms][,MaxTms][,TmsName]
                        [,AuthentName][,ScriptLoc]
                        [,DefaultCharset][,DefaultMimetype]
                        [,Expires]
```

E.5.2 필수항목

Server Group name = <string>

Size : 31 자 이내.

Server Group 에 대한 논리적인 이름으로써 SVRGROUP 절 내에서 유일한 값이어야 한다. SVRGROUP 절 이름은 SERVER 절의 SVGNAME 항목에서 사용된다.

NODENAME = <string>

Server Group 이 존재하는 Node 를 정의한다. 사용되는 NODENAME 은 NODE 절에서 정의한 Node 이름이어야 하며, Node 이름은 유닉스 명령어 "`uname -n`"을 이용해서 확인해 볼 수 있다.

SVRTYPE = <string>

Server Group 의 속성, 즉 어떠한 Service 를 제공하는가를 명시한다. Server 타입으로 HTML, CGI, JSV, WEBSTD, TPSTD, SSI 등을 명시할 수 있다.

E.5.3 선택항목

VhostName = <string>

Server Group 이 virtual host 에 속할 경우 소속된 호스트 이름을 적어준다.

Cousin = <literal>

multi-node 환경에서는 반드시 설정해 주어야 하는 항목이다. 다른 그룹 이름을 지정해 주면 그룹별로 서로 서버 프로세스를 공유하게 되어 지정한 그룹에는 서버 프로세스를 등록하지 않아도 해당 프로세스가 서비스를 처리할 수 있게 된다. 같은 Node 이거나 다른 Node 에 있는 그룹인 경우 모두 연속하여 지정 할 수 있다.

Backup = <literal>

장애대처 방안을 위한 항목이다. 백업 되어야 할 그룹 이름을 지정하면 장애 시 중단 없이 Service 를 수행할 수 있다.

Load = <numeric>

이 항목은 부하 분산을 위하여 제공되며 부하 분산 방법을 지정한다.

0: Dynamic Load Balancing 을 의미하며, 한 노드에서 더 이상 사용자 요구를 처리할 여력이 없다면 다른 노드에서 처리하도록 사용자 요구를 다른 노드로 넘긴다.

1~255: Rule based routing 을 의미하며, 분산을 하고자 하는 그룹들에 대하여 각각 Load 값을 지정하면 지정한 비율대로 부하를 분산할 수 있다. 1 부터 255 사이의 값을 사용할 수 있다.

AppDir = <literal>

JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

UsrLogDir = <literal> (WEBTOBDIR)/log/usrlog

사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

EnvFile = <string>

지정 그룹에 속한 Server 들에게 환경 변수로 값을 전달하고자 할 때나, 같은 Node 에 복수 개의 동종 데이터베이스 연동이 필요한 경우 지정한다.

Logging = <literal>

각 Server 그룹에 속한 Service 들이 수행되면서 이에 대한 기록을 남길 수 있다.

DBName = <string>

TMS 로 설정이 되는 경우 지정한다. 데이터베이스 고유의 이름을 지정하면 된다.

OpenInfo = <literal>

이 항목은 데이터베이스와 연결을 하기 위해 TMS 가 필요할 때 정의한다. 데이터 베이스로 연결을 초기화하고 각 데이터베이스에서 제공되는 문법으로 정의 한다.

CloseInfo = <literal>

이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우 지정한다. 데이터베이스와 연결을 끊기 위한 것으로 각 데이터베이스에서 제공되는 문법으로 정의한다.

<주의> INFORMIC database 는 특정한 경우이므로 주의한다.

TmsName = <string>

TMSNAME 은 해당 서버 그룹의 데이터베이스 관리를 담당할 TMS 프로세스 이름을 정의한다.

데이터베이스의 open 정보(OPENINFO 항목)를 등록한 경우에는 반드시 TMSNAME 에 대한 정의가 필요하다. TMS 프로세스는 데이터베이스와 관련된 시스템에서 해당 서버 그룹의 데이터베이스 관리를 담당하기 위하여 반드시 필요하다. 그러므로 Tmax 시스템에서 데이터베이스를 관리하도록 하기 위해서는 데이터베이스의 Open/Close 정보를 등록하고, 서버 그룹별로 반드시 TMSNAME 을 정의하여 TMS 프로세스를 기동 시켜야 한다.

TMS 프로세스는 데이터베이스와 연동하는 XA 로 업무(transaction)를 처리하는 트렌잭션 매니저이다. 이 프로세스는 $TMAXDIR/lib 에 있는 libtms.a 와 SVRGROUP 절에 있는 DBNAME 의 데이터베이스 라이브러리를 연결하여 생성된다.

MinTms = <numeric> (크기 : 1~10)

이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우, 부팅 시 기동되는 TMS 프로세스 수를 지정한다.

MaxTms = <numeric> (크기 : 1~10)

이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우, 최대 기동될 수 있는 TMS 프로세스 수를 지정한다.

ScriptLoc = <literal>

php 에 관련된 Server Group 을 설정할 경우 php 실행 모듈이 실제로 존재하는 곳의 경로를 설정하는 항목이다. (WEBTOBDIR)이하의 상대 경로를 지정해 주면 되는데 보통 "/cgi-bin/php"로 설정하고 php 실행 모듈을 이 곳에 놓는다. WebtoB 에서도 이를 권장한다.

DefaultCharset = <literal> (Default : none)

HTTP header 중에서 Content-Type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 절에서 적용되는 우선 순위는 Node < Vhost < SvrGroup < Directory 순이다.

"On"	기본 character set인 ISO-8859-1로 설정한다.
"Off"	해당 기능을 중단한다.
"_charset_"	사용자가 기술한 _charset_으로 설정한다.

DefaultMimetype = <literal> (Default : "text/html")

MIME-Type 을 결정할 수 없는 문서의 Default Content-Type 을 설정한다. Default Content-Type 은 SvrGroup, Vhost, Node 절의 순으로 결정된다.

E.5.4 SVRGROUP 절의 사용예

```
*SVRGROUP
htmlg       NODENAME = webmain, SVRTYPE = HTML
cgig        NODENAME = webmain, SVRTYPE = CGI
ssig        NODENAME = webmain, SVRTYPE = SSI
wbapg       NODENAME = webmain, SVRTYPE = WEBSTD
jsvg        NODENAME = webmain, SVRTYPE = JSV
```

E.6 SERVER 절

E.6.1 개요

SERVER 절에서는 실질적으로 제공하는 Service 들을 등록한다. JEUS Web Server 는 등록된 Service 만을 처리하기 때문에 새로운 Server 프로그램이 추가되는 경우 Server 의 환경파일에 반드시 등록하여야 한다. JEUS Web Server 가 제공하는 대부분의 Service 는 SERVER 절에서 등록이 가능하며 비즈니스 로직을 JEUS Web Server 를 통해 직접 호출하는 경우에만 SERVICE 절의 등록이 필요하다. 각각의 Server 는 위의 Server Group 절에 정의된 Service 종류에 따라 HTML, CGI, JSV 등으로 구분되며 Server Group 이름과 Process 의 가능한 개수 등이 같이 등록된다.

SERVER 절에는 다음과 같은 내용이 정의된다:

- 각 Server Process 가 속하는 Server Group.
- Server Process 의 최소 개수와 최대 개수.
- 동적 Server 기동을 위한 큐 개수.
- 재시작 가능 여부와 가능 횟수.

SERVER 절의 기본 형식은 다음과 같다:

```
Server name       SVGNAME=server-group-name
                  [,Clopt][,MinProc][,MaxProc]
                  [,UsrLogDir][,UriDir][,MaxQCount]
                  [,ASQCount][,SvrCPC]
                  [,SVRTYPE] ][,MaxRestart][,MaxRequests]
                  [,HttpOutBufSize][,HttpInBufSize]
```

E.6.2 필수항목

Servername = <string>

Size : 31 자 이내

Server 의 실행 파일 이름으로써 일반적으로 Server 이름은 유일(Unique) 해야 한다. 즉 하나의 Server 이름은 SERVER 절에 단 한번만 정의되어야 한다. 같은 이름을 중복하여 이용하면 환경 파일의 Compile 시에 Error 가 발생하게 된다.

SVGNAME = <string>

Server 가 속해 있는 Server Group 을 정의한다.

여기에 사용되는 값은 반드시 SVRGROUP 절에서 정의된 Server Group 이름이어야 한다. Server 와 SVRGROUP 절의 연결을 통해서 Server 가 어떤 Node 에서 동작할 것인지, 어떤 리소스 매니저(데이터 베이스(데이터 베이스)를 사용하는지 알 수 있으며. 해당 리소스 매니저를 열 때 필요한 파라미터를 넘겨 줄 수 있다.

E.6.3 선택항목

Clopt = <literal>

Server Process 가 기동 될 때 그 Server Process 로 전달되는 명령어 옵션이 있을 경우 이 항목에서 정의할 수 있다. 정의된 옵션들 중에 '--' 이전에 지정된 옵션들은 시스템에서 사용하고, 그 이후에 지정된 옵션들은 사용자가 자유롭게 사용할 수 있다.

MinProc = <numeric>

기본적으로 기동 될 Server Process 의 개수를 결정한다. 기존의 Client/Server 모델에서는 Client 당 Server Process 가 하나씩 기동 되는 형식으로 동작하였으나 JEUS Web Server 는 그보다 효율적인 구조를 가지고 있다. JEUS Web Server 에서는 Server Process 의 수는 일정하게 유지하고 하나의 Server Process 가 여러 개의 Client 요구를 Service 할 수 있다. MinProc 는 이러한 Server Process 의 개수를 조절하는 것으로써 운영 경험을 통해 적절한 개수를 지정할 필요가 있다. 이 MinProc 는 Server Process 의 최소 개수를 나타내는 것으로 처음 JEUS Web Server 가 Booting 될 때 시작되는 Process 의 수와 같다고 생각하면 된다.

MaxProc = <numeric>

MinProc 와 더불어 Server Process 개수를 결정하는 항목이다. MaxProc 는 MinProc 를 포함하여 추가적으로 기동 시킬 수 있는 Process 의 최대 개수이다. Server Process 는 기본적으로 JEUS Web Server Booting 시점에 위에서 정의된 MinProc 개수 만큼만 기동 되고, 부하가 높아지는 경우 MaxProc 개수까지 Server Process 가 자동적으로 기동 될 수 있다. 이 값 또한 운영 경험을 통해 적절한 개수 조정이 필요하다. 이는 특히 Web Server 의 성능에 많은 영향을 줄 수 있는 것이다. 만약 관리자가 자신의 Web Server 가 주로 HTML Service 를 위주로 한다면, 이에 대한 Service 의 MinProc 와 MaxProc 를 적당히 큰 값으로 설정하고 다른 것들을 적은 값으로 설정한다면 다른 Web

Server 들과 유사한 Hardware Overhead 를 가지면서도 좋은 성능을 낼 수 있게 된다.

UsrLogDir = <literal> (WEBTOBDIR)/log/usrlog

사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

UriDir = <literal>

지정한 Server 에 특정 URI(Uniform Resource Identifier) 값에 따라 이를 처리하는 Server 을 지정해 주는 항목이다. 특정 URI 가 입력으로 들어온 경우, 이를 지정한 Server 에서 처리하도록 mapping 시켜 놓을 수 있다. 이렇게 하면 한 Server process 에서 특정 URI 의 서비스를 전담하여 서비스 할 수 있으므로 우선순위가 매우 높게 설정되고 속도가 매우 빠르다. 주의할 점은 여기에 선언된 UriDir 은 아래의 URI 절과 ALIAS 절에 반드시 필요한 정보들이 정의되어야 한다는 점이다.

MaxQCount = <numeric>

Client 의 요청이 엄청난 폭주를 이루어 정상적인 Service 처리가 어려운 정도에 이를 경우 계속되는 Service 요청을 무시할 필요가 있다. 큐에 적체된 Client 의 요구수가 어느 정도 이상이 되면 새로 도착한 요구는 큐에 적체되지 않고 Client 에 즉시 에러로 응답한다. MaxQCount 는 이러한 적체 요구 한계를 설정한다. 즉, MaxQCount 만큼의 사용자 요구를 저장하여 기다리게 할 수 있다는 것이다. 만약 이 값이 크다면 사용자의 요구를 문제없이 처리할 수 있다는 장점이 있으나 너무 크게 하면 사용자에게 응답을 너무 늦게 줄 수도 있다는 문제점이 있다.

ASQCount = <numeric>

자동으로 Server Process 가 추가 기동 되기 위한 조건으로 큐에 쌓여진 요구 개수를 정의한다. 즉, 이 큐에 설정된 이상의 것이 적체되면 MinProc 에서 MaxProc 에 설정된 수만큼 차례대로 증가하게 된다.

SvrCPC = <numeric>

특수한 Server Process 에서 HTH Process 와 병렬 통신 채널 수를 결정하는 항목이다. Server Process 의 처리량이 아주 많아 하나의 채널로는 처리속도가 저하될 때 병렬 통신으로 처리 속도를 증가시킬 수 있다.

MaxRequests = <numeric> **(default value: 0 = unlimited)**

SERVER 절에 각 server 에 정의된 MaxRequests 값에 따라, 각 server 의 프로세서들이 그 값만큼의 사용자 request 를 처리한 후, auto-rebooting 된다. WB API 로 작성된 WEBSTD server 인 경우, ap 에 메모리관련 bug 가 있는 경우 유용하다. 서비스가 많은 경우 사용자 서비스의 연속성을 위해 MaxRequests 보다 많은 request 를 처리한 후, reboot 될 수도 있다.

MaxRestart = <numeric>

서버프로세스의 최대 재시작 가능 횟수를 결정한다

HttpOutBufSize = <numeric> **(default value: 4096)**

SERVER 절에서 정의되 Server 에 사용자의 요청에 response 를 보낼 때 사용하는 버퍼의 크기를 설정한다. Cache 기능을 사용하려면 0 으로 설정해야 한다.

HttpInBufSize = <numeric> **(default value: 8192)**

SERVER 절에서 정의되 Server 에 사용자의 request 를 받을 때 사용하는 버퍼의 크기를 설정한다.

E.6.4 SERVER 절의 사용예

```
*SERVER
html  SVGNAME = htmlg, MinProc = 1, MaxProc = 5
cgi         SVGNAME = cgig, MinProc = 1, MaxProc = 5, SvrCPC = 4
ssi   SVGNAME = ssig, MinProc = 1, MaxProc = 2
jsv1  SVGNAME = jsvg, MinProc = 1, MAXProc = 10
wbaps SVGNAME = wbapg, MinProc = 2, MAXProc = 5
```

E.7 SERVICE 절

E.7.1 개요

이 절은 JEUS Web Server 를 통해 비즈니스 로직을 바로 수행할 경우에만 설정이 필요하다. 이 경우 등록 Service 의 Server 타입은 보통 WEBSTD 로 등록되며 기존의 표준 CGI 를 구성하는 함수들을 새로 JEUS Web Server 에서 제공하는 함수들을 이용하여 새로운 Service 형태를 통하여 JEUS Web Server 를 통해 바로 수행될 수 있도록 한다. Service 의 이름을 정하고 각 Service 에 대해서는 다음과 같은 내용을 정의할 수 있다. 이는 기존 CGI 의 문제점을 획기적으로 개선한 것으로 많은 성능의 향상을 가져 올 수 있다.

Service 절에는 다음과 같은 내용이 정의된다:

- Service 가 제공되는 Server Process.
- Service 우선 순위
- Service 처리 제한 시간.

Service 절의 기본 환경 설정 형식은 다음과 같다:

```
Service name      SVRNAME=server-name
                  [,Priority][,SvcTime]
```

E.7.2 필수항목

Service Name = <string>

Size : 15 자 이내

JEUS Web Server 를 통해 바로 수행시키고자 하는 비즈니스 로직에 해당하는 Server 프로그램 내의 함수 이름 (Service 루틴명)을 명시한다.

15 자 이내의 string 으로 반드시 SERVICE 절 내에서 고유한 이름이어야 한다.

SVRNAME = <string>

해당 Service 를 제공하는 Server Process 를 지정한다. 즉, 해당 Service 루틴을 가진 Server 프로그램의 실행파일 이름이 정의된다. 앞의 Server Process 는 SERVER 절에 등록되어 있어야 한다.

E.7.3 선택항목

Priority = <numeric>

Client 의 요구를 처리하는 우선순위 값이다. 1 부터 50 까지 설정이 가능하며 숫자가 클수록 높은 우선순위를 갖는다.

SvrTime = <numeric>

Service 처리의 제한 시간이다. 즉, 각 Service 는 Service 처리가 시작되는 순간부터 끝날 때까지 지정된 SvcTime 시간 안에 처리되어야 한다. 지정 시간을 초과하면 Server Process 는 Service 를 중지하고, Client 에게 에러를 응답한다.

E.7.4 SERVICE 절의 사용예

```
*SERVICE
example            SVRNAME = webaps,
write_board        SVRNAME = webaps
```

E.8 DIRECTORY 절

E.8.1 개요

Node 내의 특정 Directory 의 속성을 정하기 위한 환경설정을 한다.

디렉토리 접근에 인증이 필요하도록 하는 AuthentName, 디렉토리 안의 파일 확장명을 설정하는 ForceMimetype, DefaultMimetype 등이 있으며, 디렉토리 접근 내역을 기록하는 로그를 설정할 수 있다.

Directory 절의 기본 환경설정 형식은 다음과 같다:

```
Directory name          DIRECTORY
                        [,DefaultCharset][,DefaultMimetype]
                        [,ForceMimetype]
```

E.8.2 필수항목

Directory name = <string>

Size : 31 자 이내

Directory 의 이름을 넣어주면 된다.

DIRECTORY = <literal>

설정을 적용할 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

E.8.3 선택항목

DefaultCharset = <literal> (Default : none)

HTTP header 중에서 Content-Type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 절에서 적용되는 우선 순위는 Node < Vhost < SvrGroup < Directory 순이다.

"On"	기본 character set인 ISO-8859-1로 설정한다.
"Off"	해당 기능을 중단한다.
"_charset_"	사용자가 기술한 _charset_으로 설정한다.

DefaultMimetype = <literal>

지정 디렉토리 안의 리소스가 요구되었는데 이 요청이 Server 가 인식하고 있지 않은 Mimetype 일 경우 DefaultMimetype 이 설정한 Mimetype 으로 처리하게 된다. 다시 말하면 DefaultMimetype 을 맡고 있는 Server 가 요구한 Service 를 처리하게 된다.

ForceMimetype = <literal>

지정 디렉토리 안의 모든 리소스들은 ForceMimetype 이 정한 Mimetype 으로 처리하게 된다. 예를 들어 ForceMimetype 이 CGI 로 되어있다면 디렉토리 내의 모든 리소스들은 Client 의 요구가 있을 시 CGI 로 처리가 된다. 즉, CGI 처리를 담당하고 있는 Server 가 Service 를 처리하게 된다.

E.8.4 DIRECTORY 절의 사용예

```
*DIRECTORY
dir_test    DIRECTORY = "/usr/local/webserver/docs/vhost_docs",
            ForceMimetype = "CGI"
```

E.9 URI 절

E.9.1 개요

URI Section 은 Client 요구의 URI(Uniform Resource Identifier)값에 따라 이를 처리하는 Service 를 구분할 수 있도록 한다. 즉, 특정 URI 가 입력으로 들어온 경우, 이를 특정 Service 에서 처리하도록 할 수 있다는 것이다. 보통 URI 절은 CGI 를 이용하는 경우에 많이 이용된다. 예를 들어, 사용자가 http://www.tmax.co.kr/cgi-bin/test.cgi 와 같은 것을 호출하였을 때 이 /cgi-bin/ URI 를 CGI 등의 Service 로 정의하여 이용할 수 있다.

URI 절의 기본 환경설정 형식은 다음과 같다.

```
URI name          URI, SVRTYPE
                  [,SVRNAME][,VhostName][,Redirect]
                  [,RedirectStatus][,SVCNAME]
```

E.9.2 필수항목

URI name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

URI = <string>

Service 속성을 지정할 URI 값을 쓰도록 한다.

SVRTYPE = <string>

Service 속성 즉, 지정된 URI 를 포함하는 Request 가 왔을 때 이를 처리할 Server 를 지정한다. 예를 들어 /jsv/ 라는 URI 를 포함하는 Request 에 대해 Server 타입이 JSV 인 Server 가 지정되도록 할 수 있다.

E.9.3 선택항목

SvrName = <string>

JEUS Web Server 에서는 같은 Server 타입을 갖고 있는 Service 개체들이 Server Group 과 Server 로 구분이 될 수 있고, 각각의 Server 는 Service 처리 Process 의 최소 및 최대 개수를 지정할 수 있다. URI 절에서는 처리를 담당할 Server 를 지정하여 세분화된 Service 제어를 할 수 있게 된다. SVRNAME 은 처리 담당 Server 의 이름을 지정한다.

WB API 사용 시에는 URI 절에서 SVRNAME 항목을 반드시 설정해 주어야 한다. 즉, URI 절의 선택 항목인 SVRNAME 은 WB API 의 사용 시에는 반드시 명시해 주어야 하는 필수 항목이 된다. 이 항목의 값은 SERVER 절에 있는 WB API 담당 서버 중 하나의 이름이 된다.

SvcName = <string>

JEUS Web Server 에서는 같은 Server 타입을 갖고 있는 Service 개체들이 Server Group 과 Server 로 구분이 될 수 있고, 각각의 Server 는 Service 처리 Process 의 최소 및 최대 개수를 지정할 수 있다. URI 절에서는 처리를 담당할 Server 를 지정하여 세분화된 Service 제어를 할 수 있게 된다. SVRNAME 은 처리 담당 Server 의 이름을 지정한다. WB API 사용 시에는 URI 절에서 SVRNAME 항목을 반드시 설정해 주어야 한다. 즉, URI 절의 선택 항목인 SVRNAME 은 WB API 의 사용 시에는 반드시 명시해 주어야 하는 필수 항

목이 된다. 이 항목의 값은 SERVER 절에 있는 WB API 담당 서버 중 하나의 이름이 된다.

VhostName = <string>

특정 Vhost 에 대한 URI 를 설정하거나 Vhost 마다 URI 는 동일하나 다른 SVRTYPE 을 쓰고 싶을 때 지정한다. 이 항목의 값은 VHOST 절에 정의된 Vhost 이름이 중 하나가 된다.

Redirect = <literal>

지정 URI 에 대한 요구를 다른 URI 에 Mapping 시키도록 하는 기능으로서 URI 절에 설정된 RedirectionStatus 의 값에 따라 Redirect 에 설정된 값이 http response 의 Location header 에 설정되어 사용자에게 보내지게 된다. RedirectStatus 의 값이 생략된 채로 Redirect 만 사용된 경우 그 값은 found(302)를 사용하게 된다.

RedirectStaus = <literal>

Redirect 기능 사용시 발생될 http status 설정한다. 이 값은 http response 의 Location header 에 설정되어 사용자에게 보내진다.

E.9.4 URI 절의 사용예

```
*URI
uri1        Uri = "/cgi-bin/", SVRTYPE = CGI
uri2        Uri = "/cgi/", SVRTYPE = CGI
uri3        Uri = "/test/", SVRTYPE = CGI
uri4        Uri = "/jsv/", SVRTYPE = JSV
```

E.10 ALIAS 절

E.10.1 개요

실제 Server 안의 물리적 디렉토리 경로와 URI 를 Alias 시키도록 설정할 수 있다. 즉, 어떤 특정한 URI 에 대한 요구가 들어오면 이를 실제의 물리적인 Directory 에 Mapping 시켜서 이곳에서 원하는 Resource 를 찾아 처리하게 하는 방식이다. 이는 사용자가 Document root 에 상관없이 지정할 수 있기 때문에 관리하는 입장에서 매우 편리할 것이다. 그러나, 이곳에 지정되는 물리적인 Directory 는 반드시 절대경로 이어야 한다.

ALIAS 절의 기본 환경설정 형식은 다음과 같다:

```
Alias name        URI, REALPATH
                  [,SVRTYPE][,NodeName][,VhostName]
```

E.10.2 필수항목

Alias name =<string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

URI =string

Alias 시키고자 하는 URI 를 명시한다.

REALPATH =<string>

Server 안의 물리적 Directory 의 경로명으로 절대 경로명을 명시한다.

E.10.3 선택항목

SVRTYPE = <string>

Alias 가 실행될 때 이를 수행할 Server 의 Type 을 지정할 수 있다.

NodeName = <string>

Alias 절이 적용되는 Node 를 지정한다. 즉, 이 경우엔 다른 Node 의 Directory 에 있는 것을 이용할 수 있다는 것이다.

VhostName = <string>

Alias 절이 적용되는 Virtual Host 를 제한하고 싶을 때 명시한다.

E.10.4 ALIAS 절의 사용예

```
*ALIAS
alias1      URI = "/cgi-bin/",
            RealPath = "/usr/local/webserver/cgi-bin/"
alias2      URI = "/tpsvc/",
            RealPath = "/usr/local/webserver/ap/"
```

E.11 DIRINDEX 절

E.11.1 개요

Client 가 요구한 index.html 과 같은 특정 파일이 존재하지 않는 경우, 전체 디렉토리 구조를 보여줄 수 있도록 설정한다. Indexing 하는 방식과 Icon 등을 지정할 수 있다.

DIRINDEX 절의 기본 환경설정 형식은 다음과 같다:

```
DIRINDEX name           OPTIONS
                        [,Ignore][,DefaultIcon][,Description]
                        [,HeaderFile][,TailFile][,IconExt]
```

참고: DIRINDEX 절을 사용하기 위해서 NODE 절에서 우선적으로 설정되어야 하는 항목들이 있다. DIRINDEX 절의 설정과 관계되는 NODE 절의 항목들은 다음과 같다 :

Options : "+Index"로 설정하면 Directory 정보를 보여준다.

DirIndex : DIRINDEX 이름을 적어준다.

E.11.2 필수항목

DIRINDEX name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

OPTIONS = <literal>

Indexing 하는 방식을 여러 가지 option 을 사용하여 설정할 수 있다.

FancyIndexing

JEUS Web Server 가 제공하는 Fancy Indexing 을 on/off 로 설정할 수 있다. 컬럼 Title 을 Click 함으로써 Value 별로 Entry 들을 정렬할 수 있으며 한번 더 Click 하면 정렬 순서가 바뀌게 된다

Option 을 넣는 경우에 FANCY 로 넣으면 된다.

E.11.3 선택항목

Ignore = <string>

Ignore 항목에서는 Indexing 시 필요에 따라 표시하고 싶지 않은 파일들이 Index 리스트에 나타나지 않도록 해준다. Indexing 에서 제외하고 싶은 파일의 List 를 파일명을 써주거나 Wildcard(*)를 사용하여 명시한다.

DefaultIcon = <string>

Unknown File Type 에 대한 Icon 을 설정한다.

Description = <literal>

특정한 파일에 대한 Description 을 설정한다.

HeaderFile = <string>

Indexing 의 가장 윗부분에 집어 넣을 Header 를 지정된 파일에서 읽어 올 수 있도록 한다. 파일명은 Indexing 하고 있는 Directory 에서 상대적인 경로명으로 간주한다.

TailFile = <string>

Indexing 의 가장 아래 부분에 집어 넣는 내용을 TailFile 에서 지정한 파일에서 읽어 온다.

HeaderFile 과 마찬가지로 파일명은 Indexing 하고 있는 Directory 에서 상대적인 경로명으로 간주한다.

IconExt = <literal>

파일의 extension 값에 따라 그에 맞는 Icon 으로 Indexing 을 할 수 있도록 설정한다. 즉, Icon 파일의 URL 과 Mimetype 을 매칭시킨다.

E.11.4 DIRINDEX 절의 사용예

```
*DIRINDEX
Index       Options = "FANCY"
```

E.12 LOGGING 절

E.12.1 개요

Client 의 요구 내역을 기록하는 형식을 지정한다. 접근 내역과 에러 내역이 따로 저장되며 저장 형식을 지정할 수 있다.

LOGGING 절의 기본 환경설정 형식은 다음과 같다:

```
Logging name            FILENAME,
                        FORMAT
                        [,Option][,ValidDays]
```

참고: LOGGING 절을 사용하기 위해서 NODE 절에서 우선적으로 설정되어야 하는 항목들이 있다. LOGGING 절의 설정과 관계되는 NODE 절의 항목들은 다음과 같다:

Logging: 해당 log 이름.

Errorlog: 해당 error log 이름.

E.12.2 필수항목

Logging name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

FILENAME = <literal>

로그를 저장할 파일의 경로명과 파일 이름을 설정한다.

FORMAT = <literal>

Log File 에 기록될 내용과 기록 방식을 설정한다. Format 이 따로 설정되지 않으면 JEUS Web Server 가 지원하는 default 로그 파일 형식이 사용된다.

E.12.3 선택항목

Option = <string>

Logging 방식을 Option 을 줌으로써 바꿀 수 있다.

Sync

Logging 내역이 메모리에 버퍼링 되지 않고 바로 디스크에 기록 되도록 한다. 증권업무나 은행업무 등, 사용자의 Log 를 바로 확인하여야 하는 곳에서는 이러한 Option 을 주어서 사용하면 문제시 쉽게 확인 할 수 있는 장점이 있다. 그러나 계속 I/O 가 발생하기 때문에 성능 저하가 발생할 수 있다.

ValidDays = <literal>

Log 를 하나의 파일에 저장할 것인지, 하루 단위로 다른 이름으로 저장할 것인지를 결정한다. 설정이 되지 않은 경우 기본값인 하루 단위(0 시기준)로 Log 파일을 저장하게 된다.

"1"	하루 단위(0시 기준)로 Log 파일을 저장한다.
"0" | "disable"	하나의 파일에 로그를 저장한다.

E.12.4 LOGGING 절의 사용예

```
*LOGGING
log1        Format = "DEFAULT",
            FileName = "/usr/local/webserver/log/access.log",
            Option = "sync"
log2        Format = "ERROR",
            FileName = "/usr/local/webserver/log/error.log",
            Option = "sync"
```

E.13 AUTHENT 절

E.13.1 개요

Client 의 접근을 제한하기 위한 인증과정을 유저와 그룹 단위로 통제할 수 있도록 설정한다.

AUTHENT 절의 기본 환경 설정 형식은 다음과 같다:

```
AUTHENT name            TYPE,
                        USERFILE
                        [,GroupFile]
```

E.13.2 필수항목

AUTHENT name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

TYPE = <string>

Authorization control (인가 제어) 방식을 설정한다. Basic 방식을 지원한다.

USERFILE = <literal>

Authorization 을 실행하기 위한 사용자명과 암호명이 기록되는 파일을 지정한다. JEUS Web Server 는 USERFILE 관리의 편의를 돕기 위해 wsmkpw 라는 유틸리티를 제공한다. 이 유틸리티를 사용하여 사용자명과 암호화(encrypted)된 암호명을 USERFILE 에 기록할 수 있다.

E.13.3 선택항목

GroupFile = <literal>

Authorization 을 그룹단위로 실행하기 위해 사용자명과 암호명을 기록하는 파일을 따로 지정할 필요가 있을 때 사용된다. 이 파일에는 그룹의 이름과 이에 속한 구성원의 정보가 기록된다.

E.13.4 AUTHENT 절의 사용예

```
*AUTHENT
authent1          Type = Basic,
                  UserFile = "/usr/local/webserver/bin/pwfile"
```

E.14 SSL 절

E.14.1 개요

JEUS Web Server 에서 사용할 SSL 의 기능을 설정하는 곳이다. 이곳에서 정의된 형태로 SSL 서비스를 하게 된다:

```
SSL name          CertificateFile,
                  CertificateKeyFile,
                  [,CACertificateFile][,CACertificatePath]
                  [,VerifyDepth][,VerifyClient][,FakeBasicAuth]
```

```
[,RequireSSL][,DenySSL]
[,CertificateChainFile][,RequriedCiphers]
[,RandomFile][,RandomFilePerConn]
[,PassPhraseDialog][,CryptoDevice]
```

E.14.2 필수항목

SSL name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

CertificateFile = <literal>

이것은 PEM 방식으로 인코딩된 서버의 인증서이다. 이것은 DER 규칙으로 인코딩되어 있으며, 웹 상에서 전송하기 위해서 아스키 코드 처럼 이용된다. 만일 인증 역시 암호화 된 상태라면, 비밀 번호나 문장 등을 물어볼 것이다.

CertificateKeyFile = <literal>

서버에서 쓰이는 PEM 방식으로 인코딩된 인증서의 개인키를 지정한다. 만일 키가 인증서와 함께 조합되지 않았다면 이 지시자를 이용하여 키의 위치를 지정하여야 한다. 일반적으로 JEUS Web Server 의 SSL Directory 에 가져다 두게 된다.

E.14.3 선택항목

CACertificatePath = <literal>

이 지시자를 이용하여 인증서를 저장할 디렉토리를 지정할 수 있다. 이 인증서는 받아들이기 위해 준비할 사용자의 인증서를 인증할 내용을 담고 있다. 이것은 보통 PEM 방식으로 인코딩 되어 있어야 한다.

CACertificateFile = <literal>

만약 단일 CA(Certificate Agent: 인증을 대신해 주는 기업)로 부터의 사용자 인증만 받고 싶다면 CACertificatePath 지시자가 아닌, 이 지시자를 이용하여 단일 PEM 으로 인코딩된 인증 파일을 사용해야 한다

VerifyDepth = <numeric>

실제 업무에 적용되는 경우에 인증에서 개입할 부분은 순서대로 다른 CA 에 의해서 서로를 인증하는 CA 에 관한 것이다. 이 지시자는 얼마나 깊은 레벨로 연결된 CA 들을 추적하여 인증할 것인지를 지시하는 것이다. 단 하나의 인증 CA 만 필요하다면 1 로 설정 하면 된다.

VerifyClient = <numeric>

이 지시자는 사용자에게 요청할 인증 레벨을 설정한다.

0: 아무런 인증 요청을 하지 않는다.

1: 사용자는 사용 가능한 인증을 서버에게 보여 주어야 한다.

2: 사용자는 사용 가능한 인증을 반드시 서버에게 보여 주어야 한다.

3: 사용자는 사용 가능한 인증을 보여 주어야 하며 만일 서버가 인증서를 가지고 있지 않은 상황에서는 인증서 인증 과정이 필요 없다.

FakeBasicAuth = <string>

이 지시자는 사용자측의 인증 버전인 한줄짜리 사용자 이름을 배제한 기본 인증 과정을 통해서 마치 인증한 것처럼 보여 준다. 만약 이 지시자를 VerifyClient 지시자와 함께 설정하면 결과는 로그 파일에서 볼 수 있을 것이다. 그리고 코드는 미리 정해진 비밀 번호를 추가하여 둔다

RandomFile = <literal>

이 지시자는 SSL 에서 이용할 Random Seed 를 위한 파일을 선택하는 것이다. 이 파일을 임의로 선택하면, JEUS Web Server 는 SSL 을 위한 암호를 만들 때 이 파일에서 임의의 값을 추출하여 Random Seed 를 만드는 것이다.

RandomFilePerConnection = <literal>

이 지시자는 위에서 언급한 RandomFile 을 설정할 때 실제 각 SSL 을 통한 연결을 맺을 때 몇 개의 파일을 이용하여 Random Seed 를 만들 것인지 결정하는 것이다.

CertificateChainFile = <literal> (Default : "builtin")

서버 인증서(Certificate)의 인증서 연쇄(Certificate Chain)를 구성할 때 필요한, 상위 인증기관들(CAs)의 인증서(certificate) 경로를 설정한다.

단, Client 의 인증 (authentication)사용하기 위해서는 CACertificateFile 이나, CACertificatePath 에 설정해야 한다.

PassPhraseDialog = <literal>

WebtoB 에서 SSL 사용시에 암호화된 개인키(encrypted private key) 파일들에 대한 암호문을 얻기위한 방식을 설정하는 항목입니다.

"builtin" : WebtoB 기동시 암호문을 입력할 것을 요구합니다.

"exec:/path/to/program" : WebtoB 기동시 해당 program 을 실행하고, 그 출력 결과를 암호문으로 사용합니다.

exec 로 실행되는 파일은 컴파일된 실행파일이나, 쉘 스크립트가 이용될 수 있습니다.

E.14.4 CA 명령어 사용 (Unix)

앞에서 우리는 CertificateFile 혹은 CertificateKeyFile 등의 항목을 통하여 사용자 인증을 위한 Key 및 인증서가 있는 File 의 위치를 선언할 수 있다. 이 때, 보통 외부 보안 업체에서 이 Key 를 발급 받아서 이용하는 경우도 있으나, 서버의 관리자가 이를 직접 만들어서 접속하는 사용자와 보안에 관한 정보를 직접 주고 받는 것도 가능하다. 이 때 서버는 자신이 만드는 보안에 대한 Key 및 인증서가 필요한데 이는 CA 라는 Program 을 통하여 이루어 진다. 이는 JEUS Web Server 의 실행 디렉토리 (/bin) 아래에 존재한다.

`-newcert ...`

인증서를 만드는 작업이다. 실제 서버가 인증서를 만드는 작업으로 주로 이를 이용하여 사용자에게 인증에 필요한 정보를 내보낼 것이다. 이 명령어를 통하여 새로운 인증서를 만들어 내는 경우 Private Key 와 인증서가 생성 된다. 또한 이를 만들어 내는 경우 Password 를 기입하여야 하는데, 이는 반드시 기억하여야 한다. 그 이유는 이를 이용하여 자신의 인증서를 만든 후에 JEUS Web Server 를 booting 시킬 때 이 Password 를 물어 보기 때문이다. 즉, 인증서를 제작할 때 입력한 Password 와 동일한 Password 를 기입하여야 JEUS Web Server 가 booting 가능하게 된다.

`-newca`

입력으로 들어온 파일에 대하여 정당한 것인지 확인하는 것이다.

`-newreq`

certificate request form 을 만드는 작업이다. 이는 특정한 Request Form 를 만들어 내어 이를 Verisign 등 인증서를 제작하거나 인증을 하는 기관에서 필요로 하게 된다. 즉, 외부 인증 기관에 필요한 인증 Request Form 을 만들어 내는 것이다.

`-sign`

만들어진 Key 등에 자신의 Sign 을 기입하는 것이다.

기타 SSL 에 관련된 정보를 알고 싶다면 다른 Reference 나 Site 를 참조하기 바란다. SSL 자체가 상당히 복잡하고 많은 내용을 담고 있기 때문에 이를 자세하게 설명하는 것은 불가능하므로, 가급적 다른 서적을 보기를 권한다.

E.14.5 SSL 절의 사용예

```
*SSL
ssl1   CertificateFile = "/user/webserver/ssl/newcert.pem",
       CertificateKeyFile = "/user/webserver/ssl/newcert.pem",
       RandomFile = "/user/webserver/bin/.rnd, 2048",
       RandomFilePerConnection = "/user/webserver/bin/.rnd, 512",
       VerifyClient = 0,
       VerifyDepth = 10,
       FakeBasicAuth = "on"
```

E.15 EXT 절

E.15.1 개요

Client 가 요구한 파일의 확장자명에 따라 처리 담당 Process 를 지정할 수 있다. JEUS Web Server 는 기본적인 모든 MIMETYPE 에 대한 처리 담당 Process 가 설정되어 있으나, 필요에 따른 추가적인 설정을 할 경우 이 절에서 할 수 있다.

EXT 절의 기본 환경 설정 형식은 다음과 같다:

```
EXT name                MIMETYPE,
                        SVRTYPE
                        [,SvrName][,Routing]
```

E.15.2 필수항목

EXT name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

MIMETYPE = <literal>

SVRTYPE 을 설정하고자 하는 MIMETYPE 을 기록한다.

SVRTYPE = <string>

앞에서 설정된 MIMETYPE 의 요구가 들어오면 이를 처리할 Server 타입을 정한다

E.15.3 선택항목

SvrName = <string>

특정한 Server 가 처리하도록 설정할 경우 Server 의 이름을 설정한다.

Routing = <string>

특정한 Service 가 처리하도록 설정할 경우 Service 의 이름을 설정한다.

E.15.4 EXT 절의 사용예

```
*EXT
htm   MimeType = "text/html", SVRTYPE = HTML
any   MimeType = "text/html", SVRTYPE = CGI
jsp   MimeType = "test/jsp", SVRTYPE = JSV
```

E.16 ACCESS 절

E.16.1 개요

Client 에서 접속을 시도할 때, IP address 나 network/netmask 들을 기준으로 요청의 허용/제한을 설정하고, 요청 허용/제한이 적용되는 순서를 설정할 수 있다. Access 절은 DIRECTORY/URI/EXT 절에 적용될 수 있으며, 각각에서 정의한 resource 를 허용/제한하게 된다

ACCESS 절의 기본 환경 설정 형식은 다음과 같다:

```
ACCESS name       [,Order] [,Allow] [,Deny]
```

E.16.2 필수항목

ACCESS name = <string>

Size : 31 자 이내

string 으로 사용자 임의로 정할 수 있다.

E.16.3 선택항목

Order = <literal>

Allow 와 Deny 가 적용되는 순서를 설정한다.

Allow = <literal>

요청이 허용되는 IP address 나 network/netmask 들을 설정한다.

Deny = <literal>

요청이 거절되는 IP address 나 network/netmask 들을 설정한다.

E.16.4 ACCESS 절의 사용예

```
*ACCESS
access1 Order = "allow, deny", Allow = "all"
access2 Order = "allow, deny", Allow = "211.1.1.10, 211.1.1.20"
access3 Order = "allow, deny", Allow = "211.1.1.0/255.255.255.0"
access4 Order = "allow, deny", Deny = "all"
```

E.17 EXPIRES 절

E.17.1 개요

클라이언트 요청에 따라 전송되는 서버응답 헤더의 정보를 설정한다. 특정 Mimetype 문서 전송시 서버응답 헤더내에 전송되는 문서의 expiry date 를 설정한다. 클라이언트 브라우저 프로그램이 같은 웹사이트에 재접속했을 때 이미 캐시에 해당 사이트의 웹 문서가 저장되어 있다면, 수정된 문서만을 받

으려고 할것이다. 이 때, 클라이언트의 브라우저는 웹서버에게 특정 Mimetype 과 해당 MIME 의 만료일자를 보내면서 expiry date 가 경과된 문서만을 재전송해달라고 웹서버에게 요청하게 된다. 웹서버는 클라이언트의 요청헤더속에 포함된 Mimetype 과 expiry date 를 분석하고해당 Mimetype 문서가 현재일자보다 기간이 경과되었을때는 재전송하게 되는 것이다. 이때, 클라이언트가 웹서버에게 보내는 expiry date 는 웹서버가 자신의 설정파일을 참조로 클라이언트에게 전송했던 것이다. EXPIRES 절은 바로 이 expiry date 의 설정과 관계되는 것을 정의한다.

EXPIRES 절의 기본 환경 설정 형식은 다음과 같다:

```
EXPIRES name      ExpiresTime = "<code><second>"
                  [,MimeType] [,Url]
```

E.17.2 필수항목

EXPIRES name = <string>

Size : 31 자 이내

string 으로 Expires 의 이름을 설정한다.

ExpiresTime = <literal>

아래와 같이 2 가지 방법으로 Expiry Date 를 명시할 수 있다. 선택항목 없이 필수항목 없이 필수항목만 명시한 경우, 무조건적으로 Expiry Date 가 Response Header 에 설정된다.

1) <code><seconds>

<code>	'M': file Modification date + <seconds>로 expiry date 설정 'A': Access time + <seconds>로 expiry date 설정
<second>	일반적으로 사용되는 시간 주기 (seconds) 1 hour(3600), 1day(86400), 1week(604800), 1moth(241920)

2) <base> [plus] {<num> <type>}*

<base>	‘access’ or ‘now’ : access time ‘modification’ : file modification date
[plus]	optional
<num>	integer value
<type>	‘years’, ‘months’, ‘weeks’, ‘days’, ‘hours’, ‘minutes’, ‘seconds’, ‘year’ , ‘month’, ‘week’, ‘day’, ‘hour’, ‘minute’ or ‘second’

E.17.3 선택항목

MimeType = <literal>

Response 의 MIME type 에 따라 Expiry Date 를 설정할 수 있다.

Url = <literal>

Request URL 이 일치하면 해당 Expiry Date 를 설정할 수 있다.

E.17.4 EXPIRES 절의 사용예

```
*EXPIRES
exp11   MimeType = "text/html", ExpiresTime = "A604800"
exp12   MimeType = "image/gif", ExpiresTime = "A2419200"
expdef1 ExpiresTime = "A86400"
exp21   Url = "/news/", ExpiresTime = "modification 1 days"
exp22   MimeType = "text/html",
        ExpiresTime = "access plus 1 weeks"
expdef2 ExpiresTime = "M86400"
exp31   Url = "/image/", MimeType = "image/gif",
        ExpiresTime = "access plus 1 months"
exp32   MimeType = "text/html", ExpiresTime = "M86400"
expdef3 ExpiresTime = "M86400"
```

E.18 ERRORDOCUMENT 절

E.18.1 개요

WebtoB 에서 에러 문제가 발생했을 때 다음과 같은 네 가지 방법으로 대응할 수 있다.

1. 소스코드에 정의된 에러 메시지를 출력한다.

2. 사용자가 정의한 에러 메시지를 출력한다.

3. 로컬 URL 로 재전송한다.

4. 외부 URL 로 재전송한다.

두 번째, 세 번째, 네 번째의 경우엔 ErrorDocument Section 을 설정하여 특정 HTTP Response Status Code 값에 대한 특정 Page 로 Redirect 를 시켜 줄 수 있도록 한 것이다. HTTP 401 Status Code 를 제외한 HTTP Status Code 를 모두 설정할 수 있다.

ERRORDOCUMENT 절의 기본 환경 설정 형식은 다음과 같다.

```
ERRORDOCUMENT name      STATUS
                        URL
```

E.18.2 필수항목

ERRORDOCUMENT name = <string> (크기 : 31 자 이내)

string 으로 ERRORDOCUMENT 이름을 설정한다

STATUS = <string>

HTTP Status Code 값을 설정한다.

URL = <literal>

DOCROOT 이하의 상대경로가 되거나 Client 가 해석할 수 있는 전체 경로 값을 설정해 주면 된다.

E.18.3 선택항목

E.18.4 ERRORDOCUMENT 절의 사용예

```
*ERRORDOCUMENT
Ercode1     Status = 403,
            Url = "err/403.html"
Ercode2     Status = 404,
            Url = "http://www.tmax.co.kr/404.html
```

E.19 TCPGW 절

E.19.1 개요

Client Request 를 listener 와 request 를 처리할 server 와 관련된 정보를 설정한다. 설정된 port 를 listen 하고 있다 data 가 오면 그 data 를 그대로 설정된 서버로 분배한다.사용 예로써는 Web Server 의 tcpgw 를 사용해서 JEUS 의 tcp-servlet 에 Round-robin 으로 data 를 분배해주는 경우가 있다. 그리고 방화벽 때문에 특정 port 만 open 이 가능한 경우 WebtoB 를 Proxy server 로 사용하는 것도 가능하다.

TCPGW 절의 기본 환경 설정 형식은 다음과 같다.

```
TCPGW name        Port
                  Listen
                  ServerAddress
                  [,Timeout][,ConnectTimeOut][,AccessName]
```

E.19.2 필수항목

TCPGW name = <string> (크기 : 31 자 이내)

TcpGW 의 이름을 설정한다.

Port = <literal>

Web Server 에서 Client 의 Request 를 Listen 하는 Port 를 설정한다. 여러 Port 를 동시에 정의하는 것도 가능하며, 현재는 최대 100 개까지의 Port 를 동시에 설정할 수 있다. 즉, Port 의 설정을 통해서 여러 개의 Port 가 동시에 Listen 하는 것이 가능하다.

주의: 주의할 점은 Port 항목은 Listen 항목과 동시에 운영할 수 없다.

Listen = <literal>

Web Server 에서 Client 의 Request 를 Listen 하는 IP Address 와 Port 를 설정한다. 여러 개의 IP Address 와 Port 번호를 설정하는 것도 가능하며, 현재는 최대 100 개까지의 IP Address 와 Port 를 동시에 설정 하는 것이 가능하다. 여러 개의 IP Address 를 가진 서버인 경우, 특정 IP Address 를 통한 요청만 받아들이기 위해서 사용할 수 있다.

주의: 주의할 점은 Port 항목은 Listen 항목과 동시에 운영할 수 없다.

ServerAddress = <literal>

Client 의 Request 를 처리 할 서버들의 IP Address 와 Port 를 설정한다. 여러 개의 서버 IP Address 와 Port 를 설정하는 것도 가능하며, 현재는 최대 100 개까지의 서버 IP Address 와 Port 를 동시에 설정 하는 것이 가능하다. 현재 설정된 서버들간의 분배규칙은 Round Robin 만 지원한다.

E.19.3 선택항목

TimeOut = <numeric> (Default Second : 300)

사용자가 Request 후, 해당 서버로부터 지정된 시간동안 응답이 없는 경우, 접속을 종료하기 위한 시간을 설정할 수 있다. 해당 시간을 0 으로 설정하게 되면, TimeOut 을 적용하지 않겠다는 의미이다.

ConnectTimeOut = <numeric> (Default Second : 5)

Client 의 Request 를 WebtoB TCP Gateway 가 받아서, 설정된 상대 서버로 접속을 시도할 때, 일정시간 동안 접속 요구에 대한 응답이 없으면, 다른 서버로 재분배 되도록 시간을 설정할 수 있다. 해당 시간을 0 으로 설정하게 되면, ConnectTimeOut 을 적용하지 않겠다는 의미이다.

AccessName = <string>

ACCESS 절의 ACCESS name 에 설정된 규칙에 따라 사용자 접속의 허용 여부를 결정한다.

E.19.4 TCPGW 절의 사용예

```
*TCPGW
tcpgw1      Port = "5000",
            ServerAddress = "192.168.1.20:5000, 92.168.1.21:5000"
```

```
tcpgw2      Listen = "192.168.1.10:5050",
            ServerAddress = "192.168.1.20:5050, 92.168.1.21:5050"
            TimeOut = 0
            AccessName = access1
tcpgw2      Listen = "192.168.1.10:5060",
            ServerAddress = "192.168.1.20:5060,92.168.1.21:5060"
            ConnectTimeOut = 20
            AccessName = access2

*ACCESS
access1 Order = "allow, deny", Allow = "all"
access2 Order = "allow, deny", Allow = "211.1.1.10, 211.1.1.20"
```

F WSMain.xml XML 구성 레퍼런스

F.1 소개

본 부록의 레퍼런스는 WSMain.xml(웹서버 환경 설정 파일)의 모든 XML 태그에 대해서 설명하고 있다. 이 파일의 XML Schema(xsd) 파일은 "JEUS_HOME\config\xsds\" 디렉토리의 "ws-main.xsd" 파일이다.

본 레퍼런스는 3 부분으로 나눠져있다.

a. **XML Schema(XSD)/XML Tree** : XML 설정 파일의 모든 태그 리스트를 정리했다. 각 노드 형식은 다음과 같다.

   a. 태그 레퍼런스로 빨리 찾아보기 위해서 각 태그마다 인덱스 번호(예 (11))를 붙여놓았다. 태그 레퍼런스에서는 이 번호 순서로 설명한다.

   b. Schema(xsd)에서 정의한 XML 태그명을 <tag name> 형식으로 표시한다.

   c. Schema(xsd)에서 정의한 Cardinality 를 표시한다. "?" = 0 개나 1 개의 element, "+" = 1 개 이상의 element, "*" = 0 개 이상의 element, (기호가 없음) = 정확히 1 개의 element

   d. 몇몇 태그에는 "P"문자를 붙여 놓았는데, 해당 태그는 성능에 관계되는 태그라는 것을 뜻한다. 이 태그는 설정을 튜닝할 때 사용된다.

b. 태그 레퍼런스 : 트리에 있는 각 XML 태그를 설명한다.

   1. **Description** : 태그에 대한 간단한 설명.

   2. **Value Description** : 입력하는 값과 타입.

   3. **Value Type** : 값의 데이터 타입. 예 : String

   4. **Default Value** : 해당 XML 을 사용하지 않았을 때 기본적으로 사용되는 값.

5. **상수** : 이미 정해져 있는 값.
6. **Example** : 해당 XML 태그에 대한 Example.
7. **Performance Recommendation** : 성능 향상을 위해서 추천되는 값.
8. **Child Elements** : 자신의 태그 안에 사용되는 태그

c. **Example XML** 파일 : "WSMain.xml"에 대한 완전한 예제

F.2 XML Schema(XSD)/XML Tree

```
(1) <ws-engine>
      (2) <domain>
            (3) <name>
            (4) <node>+
                  (5) <name>
                  (6) <webtob-dir>
                  (7) <shmkey>
                  (8) <docroot>
                  (9) <tcpgw-name>*
                  (10) <hostname>?
                  (11) <nodename>?
                  (12) <hth>? P
                  (13) <node-port>? P
                  (14) <jsv-port>?
                  (15) <rac-port>? P
                  (16) <client-check-interval>? P
                  (17) <cache-use>?
                        (18) <cache-entry> P
                        (19) <cache-size>? P
                        (20) <cache-refresh-html>? P
                        (21) <cache-refresh-dir>? P
                  (22) <max-user>?
                  (23) <txlog-dir>?
                  (24) <syslog-dir>?
                  (25) <language-priority>*
                  (26) <ip-permission>? P
                  (27) <listen-backlog>? P
```

```
        (28) <send-buffer-size>?
        (29) <limit-request-body>?
        (30) <limit-request-fields>? P
        (31) <limit-request-fieldsize>? P
        (32) <limit-request-line>? P
        (33) <server-tokens>? P
        (34) <ipc-base-port>? P
        (35) <user>?
        (36) <group>?
        (37) <admin>?
        (38) <method>*
        (39) <port>*
        (40) <listen>*
        (41) <ssl-use>?
              (42) <ssl-name>
              (43) <ssl-flag>? P
        (44) <app-dir>?
        (45) <usrlog-dir>?
        (46) <icon-dir>?
        (47) <user-dir>?
        (48) <env-file>?
        (49) <index-name>*
        (50) <dir-index-name>?
        (51) <options>*
        (52) <error-document-name>*
        (53) <logging>*
        (54) <error-log>?
        (55) <node-type>? P
        (56) <default-charset>?
        (57) <default-mimetype>? P
        (58) <expires-name>*
        (59) <service-order>? P
        (60) <keep-alive-use>?
              (61) <keep-alive> P
              (62) <keep-alive-timeout>? P
              (63) <keep-alive-max>? P
        (64) <timeout>? P
(65) <vhost>*
        (66) <name>
```

```
(67) <node-name>
(68) <docroot>
(69) <user>?
(70) <group>?
(71) <hostname>?
(72) <host-alias>*
(73) <admin>?
(74) <method>*
(75) <port>*
(76) <listen>*
(77) <ssl-use>?
      (78) <ssl-name>
      (79) <ssl-flag>? P
(80) <app-dir>?
(81) <usrlog-dir>?
(82) <icon-dir>?
(83) <user-dir>?
(84) <env-file>?
(85) <index-name>*
(86) <dir-index-name>?
(87) <options>*
(88) <error-document-name>*
(89) <logging>*
(90) <error-log>?
(91) <node-type>? P
(92) <default-charset>?
(93) <default-mimetype>? P
(94) <expires-name>*
(95) <service-order>? P
(96) <keep-alive-use>?
      (97) <keep-alive> P
      (98) <keep-alive-timeout>? P
      (99) <keep-alive-max>? P
(100) <timeout>? P
(101) <svrgroup>+
      (102) <name>
      (103) <svr-type>
      (104) <node-name>?
      (105) <vhost-name>?
```

```
        (106) <cousin>*
        (107) <backup>*
        (108) <load>?
        (109) <app-dir>?
        (110) <usrlog-dir>?
        (111) <db-name>?
        (112) <open-info>?
        (113) <close-info>?
        (114) <min-tms>?
        (115) <max-tms>?
        (116) <tms-name>?
        (117) <env-file>?
        (118) <authent-name>?
        (119) <logging>*
        (120) <script-location>?
        (121) <default-charset>?
        (122) <default-mimetype>?
        (123) <expires-name>*
(124) <server>+
        (125) <name>
        (126) <svg-name>?
        (127) <cl-opt>?
        (128) <min-proc>?
        (129) <max-proc>?
        (130) <userlog-dir>?
        (131) <uri-dir>?
        (132) <maxq-count>? P
        (133) <asq-count>? P
        (134) <max-restart>? P
        (135) <svr-cpc>? P
        (136) <svr-type>?
        (137) <http-out-buf-size>? P
        (138) <http-in-buf-size>? P
        (139) <max-requests>? P
(140) <service>*
        (141) <name>
        (142) <priority>? P
        (143) <svc-time>? P
(144) <directory>*
```

```
        (145) <name>
        (146) <dir-path>
        (147) <default-charset>?
        (148) <default-mimetype>?
        (149) <force-mimetype>?
(150) <uri>*
        (151) <name>
        (152) <uri-def>
        (153) <svr-type>
        (154) <svr-name>?
        (155) <svc-name>?
        (156) <redirect>?
        (157) <redirect-status>?
        (158) <vhost-name>?
(159) <alias>*
        (160) <name>
        (161) <uri-def>
        (162) <real-path>
        (163) <svr-type>?
        (164) <node-name>?
        (165) <vhost-name>?
(166) <dirindex>*
        (167) <name>
        (168) <option>
        (169) <ignore>*
        (170) <default-icon>?
        (171) <description>?
        (172) <header-file>?
        (173) <tail-file>?
        (174) <icon-ext>?
(175) <logging>*
        (176) <name>
        (177) <format>
        (178) <file-name>
        (179) <valid-days>?
        (180) <option>?
(181) <access>*
        (182) <name>
        (183) <order>?
```

```
      (184) <allow>*
      (185) <deny>*
(186) <authent>*
      (187) <name>
      (188) <type>
      (189) <user-file>
      (190) <group-file>
(191) <ext>*
      (192) <name>
      (193) <mimetype>?
      (194) <svr-type>?
      (195) <svr-name>?
      (196) <charset>?
(197) <ssl>*
      (198) <name>
      (199) <certificate-file>
      (200) <certificate-key-file>
      (201) <CA-certificate-path>?
      (202) <CA-certificate-file>?
      (203) <certificate-chain-file>?
      (204) <verify-depth>?
      (205) <verify-client>?
      (206) <fake-basic-auth>?
      (207) <required-ciphers>?
      (208) <random-file>?
      (209) <random-file-per-conn>?
      (210) <pass-phrase-dialog>?
      (211) <crypto-device>?
(212) <errordocument>*
      (213) <name>
      (214) <status>
      (215) <url>?
(216) <expires>*
      (217) <name>
      (218) <mimetype>?
      (219) <url>?
      (220) <expires-time>?
(221) <tcpgw>*
      (222) <name>
```

```
            (223) <port>+
            (224) <listen>+
            (225) <server-address>+
            (226) <timeout>? P
            (227) <connect-timeout>? P
            (228) <access-name>?
            (229) <schedule>? P
      (230) <nlive-inq>? P
```

F.3 Element Reference

(1) <ws-engine>

Description	JEUS Web Server 의 내용을 설정하는 최상위 레벨 테그
Child Elements	(2)domain

(2) <ws-engine> <domain>

Description	독립적인 JEUS Web Server 시스템의 전반적인 환경에 대해 정의하는 부분이다.
Child Elements	(3)name (4)node+ (65)vhost* (101)svrgroup+ (124)server+ (140)service* (144)directory* (150)uri* (159)alias* (166)dirindex* (175)logging* (181)access* (186)authent* (191)ext* (197)ssl* (212)errordocument* (216)expires* (221)tcpgw* (230)nlive-inq?

(3) <ws-engine> <domain> <name>

Description	DOMAIN 의 이름을 설정한다. DOMAIN 의 이름은 호스트 이름과 함께 암호화 되어 JEUS Web Server 의 License 확인에 사용된다.
Value Description	string 형식으로 31 자 이내로 이름을 설정한다.
Value Type	len32Type

(4) <ws-engine> <domain> <node>

Description	Web Server 를 이루는 각 Node 들에 대한 구체적인 환경을 설정한다.

Child Elements

```
(5)name
(6)webtob-dir
(7)shmkey
(8)docroot
(9)tcpgw-name*
(10)hostname?
(11)nodename?
(12)hth?
(13)node-port?
(14)jsv-port?
(15)rac-port?
(16)client-check-interval?
(17)cache-use?
(22)max-user?
(23)txlog-dir?
(24)syslog-dir?
(25)language-priority*
(26)ip-permission?
(27)listen-backlog?
(28)send-buffer-size?
(29)limit-request-body?
(30)limit-request-fields?
(31)limit-request-fieldsize?
(32)limit-request-line?
(33)server-tokens?
(34)ipc-base-port?
(35)user?
(36)group?
(37)admin?
(38)method*
(39)port*
(40)listen*
(41)ssl-use?
(44)app-dir?
(45)usrlog-dir?
(46)icon-dir?
(47)user-dir?
(48)env-file?
(49)index-name*
(50)dir-index-name?
(51)options*
(52)error-document-name*
(53)logging*
(54)error-log?
(55)node-type?
(56)default-charset?
(57)default-mimetype?
(58)expires-name*
(59)service-order?
(60)keep-alive-use?
(64)timeout?
```

(5) <ws-engine> <domain> <node> **<name>**

Description

Node 의 물리적인 이름으로 Node 이름을 설정한다. 즉, 실제 등록된 Host 의 이름을 말한다. 예를 들어 UNIX 의 경우 “uname -n” 명령으로 각 Host 의 이름을 확인할 수 있다. 또한 논리적인 node name 을 사용할 수 도 있다. 이 경우 node 명이 “/etc/hosts” 파일에 등록되어 있지 않아도 된다. 하지만 논리적인 name 을 사용할 경우에는 nodename element 에 기술한 이름을 사용해야 한다. 하나의 Domain 은 하나 이상의 Node 로 이루어지므로, NODE 절에는 최소한

하나 이상의 Node 이름이 정의되어야 한다.

Value Description string 으로 정의한다. 31 자를 초과하는 이름을 사용할 경우에는 컴파일 에러는 나지 않으나 내부적으로 31 자까지만 인식한다

Value Type len32Type

(6) <ws-engine> <domain> <node> **<webtob-dir>**

Description JEUS Web Server 가 설치되어 있는 홈 Directory 의 절대 경로명이다. 경로명은 환경 변수 WEBTOBDIR 과 동일한 값이 정의 되어야 한다. JEUS Web Server 관련 작업은 JEUS Web Server 디렉토리 하에서 모두 이루어진다

Value Type len256Type

(7) <ws-engine> <domain> <node> **<shmkey>**

Description 공유 메모리 세그먼트(Shared Memory Segment)를 가리키는 값이다. JEUS Web Server 를 이루는 Process 들이, 서로 정보를 공유하기 위한 공유 메모리 Key 값을 정의할 필요가 있다. 공유 메모리 Key 값을 정의하기 전에 이 Key 값들이, 다른 프로그램 또는 다른 업무에서 사용되는지 반드시 확인해야 한다. 그렇지 않으면 JEUS Web Server 가 Booting 시에 이 프로그램과 충돌을 일으켜서 실행이 되지 않는다.

Value Description 현재 JEUS Web Server 에서 정의되는 Shared Memory 의 Key 값은 최소 32768 에서 최대 262143 까지 이다.

Value Type long

(8) <ws-engine> <domain> <node> **<docroot>**

Description JEUS Web Server 가 Web 을 통해 Service 하는 모든 문서를 포함하는 루트 디렉토리의 경로이다. 즉, JEUS Web Server 는 DOCROOT 가 지정한 디렉토리를 최상위로 하여 문서를 Service 하게 된다. Client 가 요구한 URL 은 DOCROOT 의 경로 뒤에 추가되어 ,실제 경로명을 이루게 된다. JEUS Web Server 는 이 경로를 가지고 파일에 접근하게 된다.

Value Type len256Type

(9) <ws-engine> <domain> <node> **<tcpgw-name>**

Description TcpGW 의 이름을 설정한다.

Value Type len256Type

(10) <ws-engine> <domain> <node> **<hostname>**

Description 이 항목을 설정하면 Http Response Header 의 host name Field 에 기록을 남겨준다.

Value Type len128Type

Example JEUS Web Server 가 설치된 machine 의 Domain name 을 "www.tmaxsoft.com"과 같이 넣어주면 된다.

(11) <ws-engine> <domain> <node> **<nodename>**

Description 이 항목의 값으로는 해당 머신의 HostName 을 적어 주면 되는데,

특별히 $(NODENAME)이라고 적어주면, Web Server 가 자동적으로 해당 머신의 HostName 을 적용하게 된다. 이 NodeName 항목이 추가됨으로해서 Virtual Node 개념이 추가 되었다.

Value Description 그동안 노드명의 제약사항이였던 31 자 이상의 이름이나, 한글 또는 공백문자가 포함된 HostName 도 사용할 수 있다.

Value Type len128Type

(12) <ws-engine> <domain> <node> **<hth>**

Description JEUS Web Server 에서 가장 중요한 역할을 담당하고 있는 HTH (HTTP Request Handler) Process 의 개수를 설정한다. HTH 는 실질적으로 Client Browser 와 JEUS Web Server 내부 Service Process 사이를 중계하는 Process 이다. 즉, Client 의 요청을 받아 Service 를 받을 수 있도록 적당한 Process 에 넘겨주고, 다시 처리된 결과를 수신하여 Client 에게 되돌려 준다.

Value Type int

Default Value 1

(13) <ws-engine> <domain> <node> **<node-port>**

Description Node 간 연결 Port 번호를 설정한다. Node 간 통신은 이 Port 번호를 통하여 이루어 진다. JEUS Web Server 를 여러 Node 에 설치하였을 때, 각 Node 에 있는 JEUS Web Server 는 서로 연결을 맺고 정보를 공유한다. 따라서 이 Port 는 Multi Node 에 JEUS Web Server 를 설치한 경우 반드시 필요하다.

Value Type int

Default Value 7777

(14) <ws-engine> <domain> <node> **<jsv-port>**

Description JEUS Web Server 와 Java Servlet 수행 Server 간의 연결 Port 번호이다.

Value Type int

(15) <ws-engine> <domain> <node> **<rac-port>**

Description JEUS Web Server 가 여러 Node 로 구성되어 있을 경우, Node 관리 차원에서 Node 간 통신을 위한 Port 번호를 정의한다. 위의 NodePort 와는 달리 이것은 관리 Process 중 하나인 wsracd Daemon 에서 사용하는 Port 번호이다.

Value Type int

Default Value 3333

(16) <ws-engine> <domain> <node> **<client-check-interval>**

Description WebtoB Server 에서 접속한 Client 가 살았나 죽었나를 확인할 필요가 있을 때 설정하는 항목이다. 항목 시간 간격을 설정함으로써 WebtoB 는 Client 의 생존 여부를 주어진 시간 간격마다 확인하고 죽었을 경우 Web Server 쪽에서 접속을 끊는다.

Value Description milisecond 단위

Value Type long

Default Value 30000

(17) <ws-engine> <domain> <node> **<cache-use>**

Description cache 기능 사용에 대한 설정을 한다.

Child Elements
```
(18)cache-entry
(19)cache-size?
(20)cache-refresh-html?
(21)cache-refresh-dir?
```

(18) <ws-engine> <domain> <node> <cache-use> **<cache-entry>**

Description Cache 의 총 Hashing Key 엔트리 개수를 설정한다. Web Server 에서는 Hashing 방식을 이용하기 때문에 이의 값에 따라서 Cache 기능의 성능이 영향을 받게 된다. 만약 이 값이 적게 설정되면 Hashing Key 값이 적게 되어 Key 는 쉽게 찾지만, 각 Key 에서의 값을 찾는 문제가 발생하고 Key 값이 많게 되면 다양한 Key 에 대한 값이 나오지만 각 Entry 가 적게 되어 쉽게 찾을 수 있다.

Value Type int

Default Value 128

(19) <ws-engine> <domain> <node> <cache-use> **<cache-size>**

Description Web Server 는 Server 내부 Caching 기능을 지원한다. 따라서 많은 Web Application 이 용이하게 이루어 질 수 있다. 예를 들어 세션 정보를 관리해야 하는 경우 기존의 방식처럼 쿠키를 쓰지 않고 Web Server 가 내부적으로 제공하고 있는 Cache 에 정보를 저장함으로써 더 편리하고 효율적인 작업을 수행할 수 있다. 이러한 내부 Cache 의 사이즈를 필요에 따라 조정할 수 있다.

Value Description 여기서는 Cache 의 한 엔트리의 크기로서 기본단위는 Kbyte 이다.

Value Type int

Default Value 128

(20) <ws-engine> <domain> <node> <cache-use> **<cache-refresh-html>**

Description html file 에 대한 cache refresh time 을 설정한다.

Value Description second 단위

Value Type int

Default Value 0

(21) <ws-engine> <domain> <node> <cache-use> **<cache-refresh-dir>**

Description dirindex 에 대한 cache refresh time 을 설정한다.

Value Description second 단위

Value Type int

Default Value 0

(22) <ws-engine> <domain> <node> **<max-user>**

Description 서버 프로세스에 속한 노드의 최대 동시 접속자 수를 설정한다.

Value Type token

(23) <ws-engine> <domain> <node> **<txlog-dir>**

Description 트랜젝션 관련 로그를 기록하는 디렉토리의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(24) <ws-engine> <domain> <node> **<syslog-dir>**

Description 시스템 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다. 시스템 메시지란 wsm, htl, hth 등 JEUS Web Server 기동의 핵심 Process 들이 발생한 메시지들과 시스템 내부적으로 발생한 메시지들을 일컫는다. 이 항목은 오직 SysLogDir 를 재정의하여 사용하기 위한 것으로, 이 항목을 설정하지 않는 경우에는 Default Path 에 Log 가 남는다. Default Path 를 수정하여 사용할 경우 주의를 요한다.

Value Type len256Type

(25) <ws-engine> <domain> <node> **<language-priority>**

Description 접속 Client 가 사용 언어를 지정하지 않았을 경우 Server 쪽에서 지정된 언어 순서대로 Multiview request 등의 처리가 이루어지도록 한다.

Value Type len256Type

(26) <ws-engine> <domain> <node> **<ip-permission>**

Description IPCPERM(inter-process communication permission mask)은 Web Server 시스템에 대한 관리자가 아닌 개발자와 같은 다른 사용자가 wsdown 이나 특정 프로세스를 기동 및 종료를 할 수 있으며 이를 wsadmin 를 통해 확인할 수 있도록 한다. Unix 시스템 환경 하에서는 관리자 개인이나 그룹,기타에게 각각 파일 접속 제어(판독기능/기록기능/수행기능)를 지정 할 수 있다. 즉 default 인 경우에는 다른 사용자들은 위의 기능을 사용할 수 없으나 IPCPERM 이 0777 인 경우에는 다른 사용자들은 위의 모든 기능을 사용할 수 있다.

Value Type int

Default Value 700

(27) <ws-engine> <domain> <node> **<listen-backlog>**

Description 접속을 기다리는 큐(queue)의 길이를 제한하는 것으로, 보통은 거의 필요하지 않지만 서버가 대량의 접속 시도를 한꺼번에 날려주는 TCP SYN 해킹을 당하고 있다면 유용하게 사용될 수 있을 것이다. 만일, Request 가 Web Server 가 처리할수 있는 것보다 빠르게 요구되어지면, Listen queue 는 Overflow 가 되는데, 이때 추가적인 Request 는 Listen queue 에 여유가 생길때까지 OS 에 의해 거절되어진다.

Value Type token

Default Value 511

(28) <ws-engine> <domain> <node> **<send-buffer-size>**

Description TCP 전송 Buffer 의 크기를 설정하는 것으로, 이 항목을 이용하면 특정한 환경에서 동작 속도를 향상시킬 수 있다.

Value Type token

(29) <ws-engine> <domain> <node> **<limit-request-body>**

Description 클라이언트의 요청시 HTTP 프로토콜을 통해 서버가 제공할 수 있는 Request Body 크기를 바이트 단위로 정의한다.

Value Type int

(30) <ws-engine> <domain> <node> **<limit-request-fields>**

Description 클라이언트의 요청시 허용되는 HTTP Request header field 의 수를 설정한다.

Value Description Request Header Field 의 수

Value Type token

Default Value 100

(31) <ws-engine> <domain> <node> **<limit-request-fieldsize>**

Description 클라이언트의 요청시 허용되는 각 HTTP Request header field 의 크기를 설정한다.

Value Description 바이트(bytes)

Value Type token

Default Value 8190

(32) <ws-engine> <domain> <node> **<limit-request-line>**

Description 클라이언트의 요청시 허용되는 HTTP Request line 의 최대 크기를 설정한다.

Value Description 바이트(bytes)

Value Type token

Default Value 8190

(33) <ws-engine> <domain> <node> **<server-tokens>**

Description HTTP 응답 헤더의 Server 에 관한 정보를 어떻게 다룰지 결정한다.

Value Type len256Type

Default Value MIN

(34) <ws-engine> <domain> <node> **<ipc-base-port>**

Description Web Server 에서 내부 프로세스간 IPC 통신을 하기 위해서 해당 포트를 설정한다. 현재 Windows 에서만 지원되며, 이전 버전에서는 환경변수에 WEBTOB_WINDOWS_PORT 를 설정함으로써 같은 기능을 제공하며, 두 가지가 같이 설정이 된 경우에는 환경변수에

설정된 Port 가 우선순위를 갖는다.

Value Type int

Default Value 6666

(35) <ws-engine> <domain> <node> **<user>**

Description 설정된 User 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Value Type len32Type

(36) <ws-engine> <domain> <node> **<group>**

Description 설정된 Group 의 권한으로 JEUS Web Server 는 요구를 수행하게 된다. Client 요구 실행을 위해 따로 Group 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Value Type len32Type

(37) <ws-engine> <domain> <node> **<admin>**

Description 관리자의 정보를 나타낸다. 여기에는 관리자에게 연락할 수 있는 E-mail 주소를 설정할 수 있다.

Value Type len128Type

(38) <ws-engine> <domain> <node> **<method>**

Description Client 가 보내는 Request Method 에 대한 정의를 할 수 있다. 이 때 기본적으로 GET, POST, HEAD 등이 있어 이를 기본적으로 지원하고 만약 이들 중 특정 Method 를 쓰고 싶지 않은 경우 ? Option 을 이용하여 제거할 수 있다.

Value Type len256Type

(39) <ws-engine> <domain> <node> **<port>**

Description JEUS Web Server 의 HTTP Listener 포트 번호를 설정한다. 이는 기본적으로 Web Service 를 하기 위해서는 반드시 필요한 설정으로, 이 항목은 반드시 설정해야 한다.

Value Type len1024Type

(40) <ws-engine> <domain> <node> **<listen>**

Description JEUS Web Server 가 Booting 될 때, 원하는 IP Address 에서 연결을 맺도록 할 수 있다. 즉, 여러 개의 IP Address 를 가진 Server 에서 자신이 원하는 IP Address 에서만 Service 를 원하는 경우, 이 값을 정할 수 있다. 이 때 여러 개의 것을 복수로 설정하는 것도 가능하다.

Value Type len1024Type

(41) <ws-engine> <domain> <node> **<ssl-use>**

Description ssl 에 대한 설정을 한다. ssl-name 과 ssl-flag 를 설정할 수 있다.

Child Elements
```
(42)ssl-name
(43)ssl-flag?
```

```
(42) <ws-engine> <domain> <node> <ssl-use> <ssl-name>
```

Description JEUS Web Server 에서 SSL 을 이용하는 경우, 이에 대한 설정을 나타내는 것이다. 이는 반드시 SSLFLAG 가 on 이 된 상태에서 적용되어야 하며 off 나 설정이 되지 않은 상황에서는 아무런 의미가 없다. 이 때 지정되는 이름은 반드시 SSL 절에서 선언이 되어 있어야 한다.

Value Type len32Type

```
(43) <ws-engine> <domain> <node> <ssl-use> <ssl-flag>
```

Description JEUS Web Server 에서 SSL 을 이용할 때 반드시 지정하여야 하는 항목이다. 이 SSLFLAG 가 Y 상태이면, 그 Node 에서 SSL 을 이용하겠다는 것이고, N 상태이면, 이용하지 않는다는 것이다. 기본 설정은 SSL 을 이용하지 않는 것으로 되어 있다

Value Type token

Default Value N

```
(44) <ws-engine> <domain> <node> <app-dir>
```

Description JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

```
(45) <ws-engine> <domain> <node> <usrlog-dir>
```

Description 사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

```
(46) <ws-engine> <domain> <node> <icon-dir>
```

Description dir index 에 사용하는 아이콘 파일의 디렉토리 이름이다.

Value Type len256Type

```
(47) <ws-engine> <domain> <node> <user-dir>
```

Description JEUS Web Server 를 통해 여러 사용자를 동시에 서비스 하려는 경우 필요하다. 이 때 들어가는 값은 각 사용자의 디렉토리의 이름이다. 이를 설정하면 JEUS Web Server 는 각 사용자의 디렉토리를 찾아서 서비스를 시작한다

Value Type len256Type

```
(48) <ws-engine> <domain> <node> <env-file>
```

Description JEUS Web Server 에서 특정 정보를 읽어 들일 필요가 있는 경우

이용된다. 즉, 어떤 파일에 변수와 변수에 대한 정보를 기록한 후, JEUS Web Server 기동 시에 이 정보를 읽어야 하는 경우 이 EnvFile 에 등록된 파일을 읽어 들인다. 즉, JEUS Web Server 가 기동 시에 이 파일을 읽어서 그 정보를 가지고 있게 된다.

Value Type len256Type

(49) <ws-engine> <domain> <node> **<index-name>**

Description Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때 기본적으로 Service 되는 파일 이름을 설정한다. 따로 설정하지 않으면 index.html 이 설정된다.

Value Type len256Type

(50) <ws-engine> <domain> <node> **<dir-index-name>**

Description 뒤의 DIRINDEX 절에서 설정하는 Directory 인덱스의 이름을 적어준다.

Value Type len32Type

(51) <ws-engine> <domain> <node> **<options>**

Description Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때의 동작을 지정한다. 보통 사용자가 특정 URI 를 보내고 디렉토리 이름만 요구 하였다면 사용자에게 Directory 의 내용을 보여주는 것이 가능하다. 물론 원하지 않는 다면 보여 주지 않을 수 있다. 기본적인 설정은 보여주지 않는 것으로 되어 있다. 이 때 '+' 나 '?' Option 들을 이용하는데, 만약 디렉토리 정보를 보여주길 원한다면 "+Index"와 같이 설정하면 된다.

Value Type len256Type

(52) <ws-engine> <domain> <node> **<error-document-name>**

Description Web Server 에서 서비스 도중 특정 HTTP Error Code 에 대한 특정 Page 로 Redirect 를 시켜 주는 것이다. HTTP 401 Status Code 를 제외한 HTTP Status Code 를 모두 설정할 수 있다.

Value Type len256Type

(53) <ws-engine> <domain> <node> **<logging>**

Description 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다. 이 이름을 가지고 이 Node 에서 그에 해당하는 Log 를 남기게 되는 것이다.

Value Type len256Type

(54) <ws-engine> <domain> <node> **<error-log>**

Description 오류 발생시 설정하는 Logging 정보 이름을 써준다. 이 이름 또한 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다.

Value Type token

(55) <ws-engine> <domain> <node> **<node-type>**

Description JEUS Web Server 는 Node 단위로 Server 에 특정한 역할을 부여할 수 있다. 예를 들어 하나의 Node 를 프락시 Server(Proxy Server)로 쓰고 싶다면 NodeType 에서 Proxy 로 설정하면 된다.

Value Type token

Default Value 1

(56) <ws-engine> <domain> <node> **<default-charset>**

Description HTTP header 중에서 content type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 section 에서 적용되는 우선순위는 NODE<VHOST<SVRGROUP<DIRECTORY 순으로 우선순위를 가진다.

Value Type len64Type

(57) <ws-engine> <domain> <node> **<default-mimetype>**

Description MIME-Type 을 결정할 수 없는 문서의 Default Content-Type 을 설정한다.

Value Type len64Type

Default Value 1

(58) <ws-engine> <domain> <node> **<expires-name>**

Description Web Server 에서의 Expire 환경을 설정한다.

Value Type len256Type

(59) <ws-engine> <domain> <node> **<service-order>**

Description HTTP 요청으로부터 해당 Server 와 Service 를 결정할때, URI 절과 EXT 절의 우선순위를 결정한다. Vhost 절에 이 항목이 설정되지 않은 경우는 Node 절에 설정된 값이나 기본값을 Vhost 가 따르게 된다.

Value Type len256Type

Default Value URI,EXT

(60) <ws-engine> <domain> <node> **<keep-alive-use>**

Description keep-alive 기능에 대한 설정을 한다. keep-alive 기능 사용 여부, keep-alive-timeout, keep-alive-max 값을 설정한다.

Child Elements
(61)keep-alive
(62)keep-alive-timeout?
(63)keep-alive-max?

(61) <ws-engine> <domain> <node> <keep-alive-use> **<keep-alive>**

Description KeepAlive 는 HTTP 1.1 스펙에 포함된 기술로, 어떤 사용자가 웹사이트에 접속할 때, 그들이 웹사이트내의 다른 웹페이지를 읽어들이기 위해 곧 다시 접속을 시도하려는 경우가 매우 많다는 것을 쉽게 예상할 수 있을 것이다. 이럴 경우에 불필요한 시간 지연이 없도록 하려면 이 항목을 지정함으로써 접속을 단절하지 않고 유지할 수 있다.

Value Type token

Default Value N

(62) <ws-engine> <domain> <node> <keep-alive-use> **<keep-alive-timeout>**

Description 하나의 Client 가 불필요하게 커넥션을 오래 잡고 있는 경우를 막기 위해서 요구간, 시간 간격이 일정 시간 이상이 되면 커넥션을 끊을 수 있도록 설정할 수 있다.

Value Description milisecond 단위

Value Type long

Default Value 60000

(63) <ws-engine> <domain> <node> <keep-alive-use> **<keep-alive-max>**

Description 보통은 하나의 Client 가 한 개 이상의 요구를 연속적으로 Server 에 요청하는 경우가 많다. 이러한 경우 매 요구마다 연결을 다시 맺어야 한다면 비효율적일 것이다. 따라서 일정 개수의 요구는 처음 커넥션을 유지한 상태로 Service 를 하고 커넥션을 끊도록 한다. 커넥션을 끊기 전에 들어주는 요구의 개수를 KeepAliveMax 에서 지정한다

Value Type int

Default Value 0

(64) <ws-engine> <domain> <node> **<timeout>**

Description 사용자가 접속을 하여 Data 를 내려 받거나, 사용자의 요구를 내려 받는 시간을 지정하는 것이 가능하다. 이 때 이 Timeout Field 를 이용하게 되는데, 이 Field 를 통해서 사용자의 최대 접속시간을 지정할 수 있다. 이는 사용자와 맺은 연결이 문제가 생겨 계속해서 의미 없는 Data 전송이 발생할 때 이를 방지하기 위해서 이용한다.

Value Description Milisecond 단위

Value Type long

Default Value 300000

(65) <ws-engine> <domain> **<vhost>**

Description Web Server 로 Virtual Hosting 이 필요한 경우, 이에 대한 환경 설정을 한다. Virtual Host 기능은 실제로는 하나의 Web Server 가 동작하지만 각기 다른 URL 로 다른 문서를 제공하도록 함으로써 마치 여러 개의 Server 가 Service 를 제공하는 것처럼 보이도록 하는 기능이다.

Child Elements

```
(66)name
(67)node-name
(68)docroot
(69)user?
(70)group?
(71)hostname?
(72)host-alias*
(73)admin?
(74)method*
(75)port*
(76)listen*
(77)ssl-use?
(80)app-dir?
(81)usrlog-dir?
(82)icon-dir?
(83)user-dir?
(84)env-file?
```

```
(85)index-name*
(86)dir-index-name?
(87)options*
(88)error-document-name*
(89)logging*
(90)error-log?
(91)node-type?
(92)default-charset?
(93)default-mimetype?
(94)expires-name*
(95)service-order?
(96)keep-alive-use?
(100)timeout?
```

`(66) <ws-engine> <domain> <vhost> <name>`

Description string 으로 사용자 임의로 정의할 수 있다.

Value Type len32Type

`(67) <ws-engine> <domain> <vhost> <node-name>`

Description Virtual Host 가 속해있는 Node 의 이름을 적어준다. 이 Node 이름은 NODE 절에 정의되어 있어야 한다.

Value Type len32Type

`(68) <ws-engine> <domain> <vhost> <docroot>`

Description 정의된 Virtual Host 가 Service 하게 될 HTML 문서가 있는 최상위 디렉토리 경로명을 적어준다.

Value Type len256Type

`(69) <ws-engine> <domain> <vhost> <user>`

Description 설정된 User 의 권한으로 JEUS Web Server 가 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Value Type len32Type

`(70) <ws-engine> <domain> <vhost> <group>`

Description 설정된 Group 의 권한으로 JEUS Web Server 가 요구를 수행하게 된다. Client 요구 실행을 위해 따로 User 이름을 설정하는 것을 권장한다. 꼭 필요하지 않다면 root 권한 설정을 피하도록 한다. 이 항목은 NT OS 에서는 지원하지 않으므로 Unix 계열의 OS 에서만 적용이 가능하다.

Value Type len32Type

`(71) <ws-engine> <domain> <vhost> <hostname>`

Description Virtual Host 에 접근할 때 유저가 사용할 호스트 이름을 적어준다. Name-based Virtual Hosting 을 하는 경우 각각의 VHOST 절에서 호스트 이름을 다르게 정의한다. 이 경우는 호스트 이름으로 Virtual Host 가 구분된다.

Value Type len128Type

(72) <ws-engine> <domain> <vhost> **<host-alias>**

Description Vhost 절에 정의된 호스트 이름 이외의 다른 HostName 으로 Alias 설정할 수 있다. 이 때 HostAlias 값 설정시 WildCard(*,?) 사용이 가능하다.

Value Type len256Type

(73) <ws-engine> <domain> <vhost> **<admin>**

Description 관리자의 정보를 나타낸다. 여기에는 관리자에게 연락할 수 있는 E-mail 주소를 설정할 수 있다.

Value Type token

(74) <ws-engine> <domain> <vhost> **<method>**

Description Client 가 보내는 Request Method 에 대한 정의를 할 수 있다. 이 때 기본적으로 GET, POST, HEAD 등이 있어 이를 기본적으로 지원하고 만약 이들 중 특정 Method 를 쓰고 싶지 않은 경우 ? Option 을 이용하여 제거할 수 있다.

Value Type len256Type

(75) <ws-engine> <domain> <vhost> **<port>**

Description JEUS Web Server 의 HTTP Listener 포트 번호를 설정한다. 이는 기본적으로 Web Service 를 하기 위해서는 반드시 필요한 설정으로, 이 항목은 반드시 설정해야 한다.

Value Type len1024Type

(76) <ws-engine> <domain> <vhost> **<listen>**

Description JEUS Web Server 가 Booting 될 때, 원하는 IP Address 에서 연결을 맺도록 할 수 있다. 즉, 여러 개의 IP Address 를 가진 Server 에서 자신이 원하는 IP Address 에서만 Service 를 원하는 경우, 이 값을 정할 수 있다. 이 때 여러 개의 것을 복수로 설정하는 것도 가능하다.

Value Type len1024Type

(77) <ws-engine> <domain> <vhost> **<ssl-use>**

Description ssl 에 대한 설정을 한다. ssl-name 과 ssl-flag 를 설정할 수 있다.

Child Elements
(78)ssl-name
(79)ssl-flag?

(78) <ws-engine> <domain> <vhost> <ssl-use> **<ssl-name>**

Description JEUS Web Server 에서 SSL 을 이용하는 경우, 이에 대한 설정을 나타내는 것이다. 이는 반드시 SSLFLAG 가 on 이 된 상태에서 적용되어야 하며 off 나 설정이 되지 않은 상황에서는 아무런 의미가 없다. 이 때 지정되는 이름은 반드시 SSL 절에서 선언이 되어 있어야 한다.

Value Type len32Type

(79) <ws-engine> <domain> <vhost> <ssl-use> **<ssl-flag>**

Description JEUS Web Server 에서 SSL 을 이용할 때 반드시 지정하여야 하는 항목이다. 이 SSLFLAG 가 Y 상태이면, 그 Node 에서 SSL 을 이용하겠다는 것이고, N 상태이면, 이용하지 않는다는 것이다. 기본 설정은 SSL 을 이용하지 않는 것으로 되어 있다

Value Type token

Default Value N

(80) <ws-engine> <domain> <vhost> **<app-dir>**

Description JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(81) <ws-engine> <domain> <vhost> **<usrlog-dir>**

Description 사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(82) <ws-engine> <domain> <vhost> **<icon-dir>**

Description dir index 에 사용하는 아이콘 파일의 디렉토리 이름이다.

Value Type len256Type

(83) <ws-engine> <domain> <vhost> **<user-dir>**

Description JEUS Web Server 를 통해 여러 사용자를 동시에 서비스 하려는 경우 필요하다. 이 때 들어가는 값은 각 사용자의 디렉토리의 이름이다. 이를 설정하면 JEUS Web Server 는 각 사용자의 디렉토리를 찾아서 서비스를 시작한다.

Value Type len256Type

(84) <ws-engine> <domain> <vhost> **<env-file>**

Description JEUS Web Server 에서 특정 정보를 읽어 들일 필요가 있는 경우 이용된다. 즉, 어떤 파일에 변수와 변수에 대한 정보를 기록한 후, JEUS Web Server 기동 시에 이 정보를 읽어야 하는 경우 이 EnvFile 에 등록된 파일을 읽어 들인다. 즉, JEUS Web Server 가 기동 시에 이 파일을 읽어서 그 정보를 가지고 있게 된다.

Value Type len256Type

(85) <ws-engine> <domain> <vhost> **<index-name>**

Description Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때 기본적으로 Service 되는 파일 이름을 설정한다. 따로 설정하지 않으면 index.html 이 설정된다.

Value Type len256Type

(86) <ws-engine> <domain> <vhost> **<dir-index-name>**

Description 뒤의 DIRINDEX 절에서 설정하는 Directory 인덱스의 이름을 적어준다.

Value Type len32Type

(87) <ws-engine> <domain> <vhost> **<options>**

Description Client 가 특정 파일 이름을 지정하지 않고 Service Directory 에 요구를 보낼 때의 동작을 지정한다. 보통 사용자가 특정 URI 를 보내고 디렉토리 이름만 요구 하였다면 사용자에게 Directory 의 내용을 보여주는 것이 가능하다. 물론 원하지 않는 다면 보여 주지 않을 수 있다. 기본적인 설정은 보여주지 않는 것으로 되어 있다. 이 때 '+' 나 '?' Option 들을 이용하는데, 만약 디렉토리 정보를 보여주길 원한다면 "+Index"와 같이 설정하면 된다.

Value Type len256Type

(88) <ws-engine> <domain> <vhost> **<error-document-name>**

Description Web Server 에서 서비스 도중 특정 HTTP Error Code 에 대한 특정 Page 로 Redirect 를 시켜 주는 것이다. HTTP 401 Status Code 를 제외한 HTTP Status Code 를 모두 설정할 수 있다.

Value Type len256Type

(89) <ws-engine> <domain> <vhost> **<logging>**

Description 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다. 이 이름을 가지고 이 Node 에서 그에 해당하는 Log 를 남기게 되는 것이다.

Value Type len256Type

(90) <ws-engine> <domain> <vhost> **<error-log>**

Description 오류 발생시 설정하는 Logging 정보 이름을 써준다. 이 이름 또한 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다.

Value Type len256Type

(91) <ws-engine> <domain> <vhost> **<node-type>**

Description JEUS Web Server 는 Node 단위로 Server 에 특정한 역할을 부여할 수 있다. 예를 들어 하나의 Node 를 프락시 Server(Proxy Server)로 쓰고 싶다면 NodeType 에서 Proxy 로 설정하면 된다.

Value Type len32Type

Default Value 1

(92) <ws-engine> <domain> <vhost> **<default-charset>**

Description HTTP header 중에서 content type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 section 에서 적용되는 우선순위는 NODE<VHOST<SVRGROUP<DIRECTORY 순으로 우선순위를 가진다.

Value Type len64Type

(93) <ws-engine> <domain> <vhost> **<default-mimetype>**

Description MIME-Type 을 결정할 수 없는 문서의 Default Content-Type 을 설정한다.

Value Type len64Type

Default Value 1

(94) <ws-engine> <domain> <vhost> **<expires-name>**

Description Web Server 에서의 Expire 환경을 설정한다.

Value Type len256Type

(95) <ws-engine> <domain> <vhost> **<service-order>**

Description HTTP 요청으로부터 해당 Server 와 Service 를 결정할때, URI 절과 EXT 절의 우선순위를 결정한다. Vhost 절에 이 항목이 설정되지 않은 경우는 Node 절에 설정된 값이나 기본값을 Vhost 가 따르게 된다.

Value Type len256Type

Default Value URI,EXT

(96) <ws-engine> <domain> <vhost> **<keep-alive-use>**

Description keep-alive 기능에 대한 설정을 한다. keep-alive 기능 사용 여부, keep-alive-timeout, keep-alive-max 값을 설정한다.

Child Elements
(97)keep-alive
(98)keep-alive-timeout?
(99)keep-alive-max?

(97) <ws-engine> <domain> <vhost> <keep-alive-use> **<keep-alive>**

Description KeepAlive 는 HTTP 1.1 스펙에 포함된 기술로, 어떤 사용자가 웹사이트에 접속할 때, 그들이 웹사이트내의 다른 웹페이지를 읽어들이기 위해 곧 다시 접속을 시도하려는 경우가 매우 많다는 것을 쉽게 예상할 수 있을 것이다. 이럴 경우에 불필요한 시간 지연이 없도록 하려면 이 항목을 지정함으로써 접속을 단절하지 않고 유지할 수 있다.

Value Type token

Default Value N

(98) <ws-engine> <domain> <vhost> <keep-alive-use> **<keep-alive-timeout>**

Description 하나의 Client 가 불필요하게 커넥션을 오래 잡고 있는 경우를 막기 위해서 요구간, 시간 간격이 일정 시간 이상이 되면 커넥션을 끊을 수 있도록 설정할 수 있다.

Value Description milisecond 단위

Value Type long

Default Value 60000

(99) <ws-engine> <domain> <vhost> <keep-alive-use> **<keep-alive-max>**

Description 보통은 하나의 Client 가 한 개 이상의 요구를 연속적으로 Server 에 요청하는 경우가 많다. 이러한 경우 매 요구마다 연결을 다시 맺어야 한다면 비효율적일 것이다. 따라서 일정 개수의 요구는 처음

커넥션을 유지한 상태로 Service 를 하고 커넥션을 끊도록 한다. 커넥션을 끊기 전에 들어주는 요구의 개수를 KeepAliveMax 에서 지정한다

Value Type int

Default Value 0

(100) <ws-engine> <domain> <vhost> **<timeout>**

Description 사용자가 접속을 하여 Data 를 내려 받거나, 사용자의 요구가 내려 받는 시간을 지정하는 것이 가능하다. 이 때 이 Timeout Field 를 이용하게 되는데, 이 Field 를 통해서 사용자의 최대 접속시간을 지정할 수 있다. 이는 사용자와 맺은 연결이 문제가 생겨 계속해서 의미 없는 Data 전송이 발생할 때 이를 방지하기 위해서 이용한다.

Value Description Milisecond 단위의 시간

Value Type long

Default Value 300000

(101) <ws-engine> <domain> **<svrgroup>**

Description Web Server 를 통해 응용 Server Process 를 접근하는 경우 Server Process 의 논리적인 연관성에 따라 이들을 그룹으로 관리할 필요가 있게 된다. 이 항목에서는 이러한 그룹에 대한 환경 설정이 이루어 진다.

Child Elements

```
(102)name
(103)svr-type
(104)node-name?
(105)vhost-name?
(106)cousin*
(107)backup*
(108)load?
(109)app-dir?
(110)usrlog-dir?
(111)db-name?
(112)open-info?
(113)close-info?
(114)min-tms?
(115)max-tms?
(116)tms-name?
(117)env-file?
(118)authent-name?
(119)logging*
(120)script-location?
(121)default-charset?
(122)default-mimetype?
(123)expires-name*
```

(102) <ws-engine> <domain> <svrgroup> **<name>**

Description Server Group 에 대한 논리적인 이름으로써 SVRGROUP 절 내에서 유일한 값이어야 한다. SVRGROUP 절 이름은 SERVER 절의 SVGNAME 항목에서 사용된다.

Value Type len32Type

(103) <ws-engine> <domain> <svrgroup> **<svr-type>**

Description Server Group 의 속성, 즉 어떠한 Service 를 제공하는가를 명시한다. Server 타입으로 HTML, CGI, JSV, WEBSTD, TPSTD, SSI 등을 명시할 수 있다.

Value Type svrElementType

Defined Value

HTML
HTML(HyperText Markup Language)

CGI
CGI(Commom Gateway Inteface)

SSI
SSI(Server Side Include)

JSV
JSV(Java SerVlet)

PHP
PHP(Personal HomePage)

WEBSTD
WEB API

(104) <ws-engine> <domain> <svrgroup> **<node-name>**

Description Server Group 이 존재하는 Node 를 정의한다. 사용되는 NODENAME 은 NODE 절에서 정의한 Node 이름이어야 하며, Node 이름은 유닉스 명령어 “uname -n”을 이용해서 확인해 볼 수 있다.

Value Type len32Type

(105) <ws-engine> <domain> <svrgroup> **<vhost-name>**

Description Server Group 이 virtual host 에 속할 경우 소속된 호스트 이름을 적어준다.

Value Type len32Type

(106) <ws-engine> <domain> <svrgroup> **<cousin>**

Description multi-node 환경에서는 반드시 설정해 주어야 하는 항목이다. 다른 그룹 이름을 지정해 주면 그룹별로 서로 서버 프로세스를 공유하게 되어 지정한 그룹에는 서버 프로세스를 등록하지 않아도 해당 프로세스가 서비스를 처리할 수 있게 된다. 같은 Node 이거나 다른 Node 에 있는 그룹인 경우 모두 연속하여 지정 할 수 있다.

Value Type len256Type

(107) <ws-engine> <domain> <svrgroup> **<backup>**

Description 장애대처 방안을 위한 항목이다. 백업 되어야 할 그룹 이름을 지정하면 장애 시 중단 없이 Service 를 수행할 수 있다.

Value Type len256Type

(108) <ws-engine> <domain> <svrgroup> **<load>**

Description 이 항목은 부하 분산을 위하여 제공되며 부하 분산 방법을 지정한다.

Value Type int

(109) <ws-engine> <domain> <svrgroup> **<app-dir>**

Description JEUS Web Server 를 통해 응용 프로그램을 바로 호출하는 경우 설정이 필요하다. 응용 프로그램의 실행 파일이 존재하는 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(110) <ws-engine> <domain> <svrgroup> **<usrlog-dir>**

Description 사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(111) <ws-engine> <domain> <svrgroup> **<db-name>**

Description TMS 로 설정이 되는 경우 지정한다. 데이터베이스 고유의 이름을 지정하면 된다.

Value Type len64Type

(112) <ws-engine> <domain> <svrgroup> **<open-info>**

Description 이 항목은 데이터베이스와 연결을 하기위해 TMS 가 필요할 때 정의한다. 데이터 베이스로 연결을 초기화하고 각 데이터베이스에서 제공되는 문법으로 정의 한다.

Value Type len256Type

(113) <ws-engine> <domain> <svrgroup> **<close-info>**

Description 이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우 지정한다. 데이터베이스와 연결을 끊기 위한 것으로 각 데이터베이스에서 제공되는 문법으로 정의 한다. <주의> INFORMIX database 은 특정한 경우이므로 주의한다.

Value Type len256Type

(114) <ws-engine> <domain> <svrgroup> **<min-tms>**

Description 이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우, 부팅 시 기동되는 TMS 프로세스수를 지정한다.

Value Type int

(115) <ws-engine> <domain> <svrgroup> **<max-tms>**

Description 이 항목은 데이터베이스와 연동하는 그룹, 즉 TMS 로 설정이 되는 경우, 최대 기동 될 수 있는 TMS 프로세스수를 지정한다.

Value Type int

(116) <ws-engine> <domain> <svrgroup> **<tms-name>**

Description TMSNAME 은 해당 서버그룹의 데이터베이스 관리를 담당할 TMS 프로세스 이름을 정의한다. 데이터베이스의 open

정보(OPENINFO 항목)를 등록한 경우에는 반드시 TMSNAME 에 대한 정의가 필요하다. TMS 프로세스는 데이터베이스와 관련된 시스템에서 해당 서버그룹의 데이터베이스 관리를 담당하기 위하여 반드시 필요하다. 그러므로 Tmax 시스템에서 데이터베이스를 관리하도록 하기 위해서는 데이터베이스의 Open/Close 정보를 등록하고, 서버그룹별로 반드시 TMSNAME 을 정의하여 TMS 프로세스를 기동 시켜야 한다. TMS 프로세스는 데이터베이스와 연동하는 XA 로 업무(transaction)를 처리하는 트렌잭션 매니저이다. 이 프로세스는 $TMAXDIR/lib 에 있는 libtms.a 와 SVRGROUP 절에 있는 DBNAME 의 데이터베이스 라이브러리를 연결하여 생성된다.

Value Type len32Type

`(117) <ws-engine> <domain> <svrgroup> <env-file>`

Description 지정 그룹에 속한 Server 들에게 환경 변수로 값을 전달하고자 할 때나, 같은 Node 에 복수 개의 동종 데이터베이스 연동이 필요한 경우 지정한다.

Value Type len256Type

`(118) <ws-engine> <domain> <svrgroup> <authent-name>`

Description Web Server 에서는 Server Group 단위의 보안을 제공한다. 만약 한 서버 그룹에 Authentication 을 사용하고자 한다면, 해당 Server Group 의 AuthentName 항목에 AUTHENT 절에 미리 선언된 Authent 이름을 설정해 주면 된다. 물론 Authentication 을 사용하려면 미리 AUTHENT 절에 Authent 이름이 선언되어 있어야 하고 Web Server 에서 제공하는 wsmkpw utility 를 통해 user 명과 password 를 설정해 주어야 한다.

Value Type len32Type

`(119) <ws-engine> <domain> <svrgroup> <logging>`

Description 뒤의 Logging 절에서 설정하는 Logging Name 을 써준다. 이 이름을 가지고 이 Node 에서 그에 해당하는 Log 를 남기게 되는 것이다.

Value Type len256Type

`(120) <ws-engine> <domain> <svrgroup> <script-location>`

Description php 에 관련된 Server Group 을 설정할 경우 php 실행 모듈이 실제로 존재하는 곳의 경로를 설정하는 항목이다.

Value Type len256Type

`(121) <ws-engine> <domain> <svrgroup> <default-charset>`

Description HTTP header 중에서 content type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 section 에서 적용되는 우선순위는 NODE<VHOST<SVRGROUP<DIRECTORY 순으로 우선순위를 가진다.

Value Type len64Type

`(122) <ws-engine> <domain> <svrgroup> <default-mimetype>`

Description 지정 디렉토리 안의 리소스가 요구되었는데 이 요청이 Server 가

인식하고 있지 않은 Mimetype 일 경우 DefaultMimetype 이 설정한 Mimetype 으로 처리하게 된다. 다시 말하면 DefaultMimetype 을 맡고 있는 Server 가 요구한 Service 를 처리하게 된다.

Value Type len64Type

(123) <ws-engine> <domain> <svrgroup> **<expires-name>**

Description Web Server 에서의 Expire 환경을 설정한다.

Value Type len256Type

(124) <ws-engine> <domain> **<server>**

Description 이 항목에서는 실질적으로 제공하는 Service 들을 등록한다. Web Server 는 등록된 Service 만을 처리하기 때문에 새로운 Server 프로그램이 추가되는 경우는 반드시 등록하여야 한다.

Child Elements

```
(125)name
(126)svg-name?
(127)cl-opt?
(128)min-proc?
(129)max-proc?
(130)userlog-dir?
(131)uri-dir?
(132)maxq-count?
(133)asq-count?
(134)max-restart?
(135)svr-cpc?
(136)svr-type?
(137)http-out-buf-size?
(138)http-in-buf-size?
(139)max-requests?
```

(125) <ws-engine> <domain> <server> **<name>**

Description Server 의 실행 파일 이름으로써 일반적으로 Server 이름은 유일(Unique) 해야 한다. 즉 하나의 Server 이름은 SERVER 절에 단 한번만 정의되어야 한다. 같은 이름을 중복하여 이용하면 환경 파일의 Compile 시에 Error 가 발생하게 된다.

Value Type len32Type

(126) <ws-engine> <domain> <server> **<svg-name>**

Description Server 가 속해 있는 Server Group 을 정의한다. 여기에 사용되는 값은 반드시 SVRGROUP 절에서 정의된 Server Group 이름이어야 한다. Server 와 SVRGROUP 절의 연결을 통해서 Server 가 어떤 Node 에서 동작할 것인지, 어떤 리소스 매니저(데이터 베이스(데이터 베이스)를 사용하는지 알 수 있으며, 해당 리소스 매니저를 열 때 필요한 파라미터를 넘겨 줄 수 있다.

Value Type len32Type

(127) <ws-engine> <domain> <server> **<cl-opt>**

Description Server Process 가 기동 될 때 그 Server Process 로 전달되는 명령어 옵션이 있을 경우 이 항목에서 정의할 수 있다. 정의된 옵션들 중에 '--' 이전에 지정된 옵션들은 시스템에서 사용하고, 그 이후에 지정된 옵션들은 사용자가 자유롭게 사용할 수 있다.

Value Type len256Type

(128) <ws-engine> <domain> <server> **<min-proc>**

Description 기본적으로 기동 될 Server Process 의 개수를 결정한다. 기존의 Client/Server 모델에서는 Client 당 Server Process 가 하나씩 기동 되는 형식으로 동작하였으나 JEUS Web Server 는 그보다 효율적인 구조를 가지고 있다. JEUS Web Server 에서는 Server Process 의 수는 일정하게 유지하고 하나의 Server Process 가 여러 개의 Client 요구를 Service 할 수 있다. MinProc 는 이러한 Server Process 의 개수를 조절하는 것으로써 운영 경험을 통해 적절한 개수를 지정할 필요가 있다. 이 MinProc 는 Server Process 의 최소 개수를 나타내는 것으로 처음 JEUS Web Server 가 Booting 될 때 시작되는 Process 의 수와 같다고 생각하면 된다.

Value Type int

(129) <ws-engine> <domain> <server> **<max-proc>**

Description MinProc 와 더불어 Server Process 개수를 결정하는 항목이다. MaxProc 는 MinProc 를 포함하여 추가적으로 기동 시킬 수 있는 Process 의 최대 개수이다. Server Process 는 기본적으로 JEUS Web Server Booting 시점에 위에서 정의된 MinProc 개수 만큼만 기동 되고, 부하가 높아지는 경우 MaxProc 개수까지 Server Process 가 자동적으로 기동 될 수 있다. 이 값 또한 운영 경험을 통해 적절한 개수 조정이 필요하다. 이는 특히 Web Server 의 성능에 많은 영향을 줄 수 있는 것이다. 만약 관리자가 자신의 Web Server 가 주로 HTML Service 를 위주로 한다면, 이에 대한 Service 의 MinProc 와 MaxProc 를 적당히 큰 값으로 설정하고 다른 것들을 적은 값으로 설정한다면 다른 Web Server 들과 유사한 Hardware Overhead 를 가지면서도 좋은 성능을 낼 수 있게 된다.

Value Type int

(130) <ws-engine> <domain> <server> **<userlog-dir>**

Description 사용자 메시지가 기록될 Directory 의 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(131) <ws-engine> <domain> <server> **<uri-dir>**

Description 지정한 Server 에 특정 URI(Uniform Resource Identifier) 값에 따라 이를 처리하는 Server 을 지정해 주는 항목이다. 특정 URI 가 입력으로 들어온 경우, 이를 지정한 Server 에서 처리하도록 mapping 시켜 놓을 수 있다. 이렇게 하면 한 Server process 에서 특정 URI 의 서비스를 전담하여 서비스 할 수 있으므로 우선순위가 매우 높게 설정되고 속도가 매우 빠르다. 주의할 점은 여기에 선언된 UriDir 은 아래의 URI 절과 ALIAS 절에 반드시 필요한 정보들이 정의되어야 한다는 점이다.

Value Type len256Type

(132) <ws-engine> <domain> <server> **<maxq-count>**

Description Client 의 요청이 엄청난 폭주를 이루어 정상적인 Service 처리가

어려운 정도에 이를 경우 계속되는 Service 요청을 무시할 필요가 있다. 큐에 적체된 Client 의 요구수가 어느 정도 이상이 되면 새로 도착한 요구는 큐에 적체되지 않고 Client 에 즉시 에러로 응답한다. MaxQCount 는 이러한 적체 요구 한계를 설정한다. 즉, MaxQCount 만큼의 사용자 요구를 저장하여 기다리게 할 수 있다는 것이다. 만약 이 값이 크다면 사용자의 요구를 문제없이 처리할 수 있다는 장점이 있으나 너무 크게 하면 사용자에게 응답을 너무 늦게 줄 수도 있다는 문제점이 있다.

Value Type int

Default Value 0

(133) <ws-engine> <domain> <server> **<asq-count>**

Description 자동으로 Server Process 가 추가 기동 되기 위한 조건으로 큐에 쌓여진 요구 개수를 정의한다. 즉, 이 큐에 설정된 이상의 것이 적체되면 MinProc 에서 MaxProc 에 설정된 수만큼 차례대로 증가하게 된다.

Value Type int

Default Value 0

(134) <ws-engine> <domain> <server> **<max-restart>**

Description 서버프로세스의 최대 재시작 가능 횟수를 결정한다.

Value Type int

Default Value 10

(135) <ws-engine> <domain> <server> **<svr-cpc>**

Description 특수한 Server Process 에서 HTH Process 와 병렬 통신 채널 수를 결정하는 항목이다. Server Process 의 처리량이 아주 많아 하나의 채널로는 처리속도가 저하될 때 병렬 통신으로 처리 속도를 증가시킬 수 있다.

Value Type int

Default Value 1

(136) <ws-engine> <domain> <server> **<svr-type>**

Description Service 속성 즉, 지정된 URI 를 포함하는 Request 가 왔을 때 이를 처리할 Server 를 지정한다. 예를 들어 /jsv/ 라는 URI 를 포함하는 Request 에 대해 Server 타입이 JSV 인 Server 가 지정되도록 할 수 있다.

Value Type svrElementType

Defined Value

HTML
HTML(HyperText Markup Language)

CGI
CGI(Commom Gateway Inteface)

SSI
SSI(Server Side Include)

JSV
JSV(Java SerVlet)

PHP
PHP(Personal HomePage)

WEBSTD
WEB API

(137) <ws-engine> <domain> <server> **<http-out-buf-size>**

Description SERVER 절에서 정의되 Server 에 사용자의 요청에 response 를 보낼 때 사용하는 버퍼의 크기를 설정한다. Cache 기능을 사용하려면 0 으로 설정해야 한다.

Value Type int

Default Value 4096

(138) <ws-engine> <domain> <server> **<http-in-buf-size>**

Description SERVER 절에서 정의되 Server 에 사용자의 request 를 받을 때 사용하는 버퍼의 크기를 설정한다.

Value Type int

Default Value 8192

(139) <ws-engine> <domain> <server> **<max-requests>**

Description SERVER 절에 각 server 에 정의된 MaxRequests 값에 따라, 각 server 의 프로세서들이 그 값만큼의 사용자 request 를 처리한 후, auto-rebooting 된다. WB API 로 작성된 WEBSTD server 인 경우, ap 에 메모리관련 bug 가 있는 경우 유용하다. 서비스가 많은 경우 사용자 서비스의 연속성을 위해 MaxRequests 보다 많은 request 를 처리한 후, reboot 될 수도 있다.

Value Type int

Default Value 0

(140) <ws-engine> <domain> **<service>**

Description 이 항목은 Web Server 를 통해 비즈니스 로직을 바로 수행할 경우에만 설정이 필요하다.

Child Elements
(141)name
(142)priority?
(143)svc-time?

(141) <ws-engine> <domain> <service> **<name>**

Description JEUS Web Server 를 통해 바로 수행시키고자 하는 비즈니스 로직에 해당하는 Server 프로그램 내의 함수 이름 (Service 루틴명)을 명시한다.

Value Description 15 자 이내의 string 으로 반드시 SERVICE 절 내에서 고유한 이름이어야 한다.

Value Type len16Type

(142) <ws-engine> <domain> <service> **<priority>**

Description Client 의 요구를 처리하는 우선순위 값이다. 1 부터 50 까지 설정이 가능하며 숫자가 클수록 높은 우선순위를 갖는다.

Value Type int

Default Value 50

(143) <ws-engine> <domain> <service> **<svc-time>**

Description Service 처리의 제한 시간이다. 즉, 각 Service 는 Service 처리가 시작되는 순간부터 끝날 때까지 지정된 SvcTime 시간 안에 처리되어야 한다. 지정 시간을 초과하면 Server Process 는 Service 를 중지하고, Client 에게 에러를 응답한다.

Value Type long

Default Value 0

(144) <ws-engine> <domain> **<directory>**

Description Node 내의 특정 Directory 의 속성을 정하기 위한 환경설정을 한다.

Child Elements
(145)name
(146)dir-path
(147)default-charset?
(148)default-mimetype?
(149)force-mimetype?

(145) <ws-engine> <domain> <directory> **<name>**

Description Directory 의 이름을 넣어주면 된다.

Value Type len32Type

(146) <ws-engine> <domain> <directory> **<dir-path>**

Description 설정을 적용할 Directory 의 경로명으로 경로명을 설정한다. 경로명은 절대 경로와 WEBTOBDIR 을 기준으로 한 상대 경로를 사용할 수 있다.

Value Type len256Type

(147) <ws-engine> <domain> <directory> **<default-charset>**

Description HTTP header 중에서 content type 에 character set 관련 파라미터가 없는 응답에 추가될 character set 의 이름을 설정한다. 여러 section 에서 적용되는 우선순위는 NODE<VHOST<SVRGROUP<DIRECTORY 순으로 우선순위를 가진다.

Value Type len64Type

(148) <ws-engine> <domain> <directory> **<default-mimetype>**

Description 지정 디렉토리 안의 리소스가 요구되었는데 이 요청이 Server 가 인식하고 있지 않은 Mimetype 일 경우 DefaultMimetype 이 설정한 Mimetype 으로 처리하게 된다. 다시 말하면 DefaultMimetype 을 맡고 있는 Server 가 요구한 Service 를 처리하게 된다.

Value Type token

(149) <ws-engine> <domain> <directory> **<force-mimetype>**

Description 지정 디렉토리 안의 모든 리소스들은 ForceMimetype 이 정한 Mimetype 으로 처리하게 된다. 예를 들어 ForceMimetype 이 CGI 로 되어있다면 디렉토리 내의 모든 리소스들은 Client 의 요구가 있을 시 CGI 로 처리가 된다. 즉, CGI 처리를 담당하고 있는 Server 가 Service 를 처리하게 된다

Value Type token

(150) <ws-engine> <domain> **<uri>**

Description Client 요구의 URI(Uniform Resource Identifier)값에 따라 이를 처리하는 Service 를 구분할 수 있도록 한다. 즉, 특정 URI 가 입력으로 들어온 경우, 이를 특정 Service 에서 처리하도록 할 수 있다는 것이다.

Child Elements
(151)name
(152)uri-def
(153)svr-type
(154)svr-name?
(155)svc-name?
(156)redirect?
(157)redirect-status?
(158)vhost-name?

(151) <ws-engine> <domain> <uri> **<name>**

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(152) <ws-engine> <domain> <uri> **<uri-def>**

Description Service 속성을 지정할 URI 값을 쓰도록 한다.

Value Type len256Type

(153) <ws-engine> <domain> <uri> **<svr-type>**

Description Service 속성 즉, 지정된 URI 를 포함하는 Request 가 왔을 때 이를 처리할 Server 를 지정한다. 예를 들어 /jsv/ 라는 URI 를 포함하는 Request 에 대해 Server 타입이 JSV 인 Server 가 지정되도록 할 수 있다.

Value Type svrElementType

Defined Value
HTML
HTML(HyperText Markup Language)

CGI
CGI(Commom Gateway Inteface)

SSI
SSI(Server Side Include)

JSV
JSV(Java SerVlet)

PHP
PHP(Personal HomePage)

WEBSTD
WEB API

(154) <ws-engine> <domain> <uri> <svr-name>

Description JEUS Web Server 에서는 같은 Server 타입을 갖고 있는 Service 개체들이 Server Group 과 Server 로 구분이 될 수 있고, 각각의 Server 는 Service 처리 Process 의 최소 및 최대 개수를 지정할 수 있다. URI 절에서는 처리를 담당할 Server 를 지정하여 세분화된 Service 제어를 할 수 있게 된다. SVRNAME 은 처리 담당 Server 의 이름을 지정한다. WB API 사용 시에는 URI 절에서 SVRNAME 항목을 반드시 설정해 주어야 한다. 즉, URI 절의 선택 항목인 SVRNAME 은 WB API 의 사용 시에는 반드시 명시해 주어야 하는 필수 항목이 된다. 이 항목의 값은 SERVER 절에 있는 WB API 담당 서버 중 하나의 이름이 된다

Value Type len32Type

(155) <ws-engine> <domain> <uri> <svc-name>

Description JEUS Web Server 에서는 같은 Server 타입을 갖고 있는 Service 개체들이 Server Group 과 Server 로 구분이 될 수 있고, 각각의 Server 는 Service 처리 Process 의 최소 및 최대 개수를 지정할 수 있다. URI 절에서는 처리를 담당할 Server 를 지정하여 세분화된 Service 제어를 할 수 있게 된다. SVRNAME 은 처리 담당 Server 의 이름을 지정한다. WB API 사용 시에는 URI 절에서 SVRNAME 항목을 반드시 설정해 주어야 한다. 즉, URI 절의 선택 항목인 SVRNAME 은 WB API 의 사용 시에는 반드시 명시해 주어야 하는 필수 항목이 된다. 이 항목의 값은 SERVER 절에 있는 WB API 담당 서버 중 하나의 이름이 된다.

Value Type len16Type

(156) <ws-engine> <domain> <uri> <redirect>

Description 지정 URI 에 대한 요구를 다른 URI 에 Mapping 시키도록 하는 기능으로서 RedirectStatus 의 값에 따라 Redirect 에 설정된 값이 http response 의 Location header 에 설정되어 사용자에게 보내지게 된다. RedirectStatus 의 값이 생략된 채로 Redirect 만 사용된 경우 그 값은 found(302)를 사용하게 된다.

Value Type len256Type

(157) <ws-engine> <domain> <uri> <redirect-status>

Description redirect 기능 사용시 발생될 http status 설정한다. 이 값은 http response 의 Location header 에 설정되어 사용자에게 보내진다.

Value Type len32Type

(158) <ws-engine> <domain> <uri> <vhost-name>

Description 특정 Vhost 에 대한 URI 를 설정하거나 Vhost 마다 URI 는 동일하나 다른 SVRTYPE 을 쓰고 싶을 때 지정한다. 이 항목의 값은 VHOST 절에 정의된 Vhost 이름이 중 하나가 된다.

Value Type len32Type

`(159) <ws-engine> <domain> **<alias>**`

Description 실제 Server 안의 물리적 Directory 경로와 URI 를 Alias 시키도록 설정할 수 있다. 즉, 어떤 특정한 URI 에 대한 요구가 들어오면 이를 실제의 물리적인 Directory 에 Mapping 시켜서 이곳에서 원하는 Resource 를 찾아 처리하게 하는 방식이다. 이는 사용자가 Document Root 에 상관없이 지정할 수 있기 때문에 관리적 입장에서 매우 편리할 것이다. 그러나, 이곳에 지정되는 물리적인 Directory 는 반드시 절대경로 이어야 한다.

Child Elements
```
(160)name
(161)uri-def
(162)real-path
(163)svr-type?
(164)node-name?
(165)vhost-name?
```

`(160) <ws-engine> <domain> <alias> **<name>**`

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

`(161) <ws-engine> <domain> <alias> **<uri-def>**`

Description Service 속성을 지정할 URI 값을 쓰도록 한다.

Value Type len256Type

`(162) <ws-engine> <domain> <alias> **<real-path>**`

Description Server 안의 물리적 Directory 의 경로명으로 절대 경로명을 명시한다.

Value Type len256Type

`(163) <ws-engine> <domain> <alias> **<svr-type>**`

Description Alias 가 실행될 때 이를 수행할 Server 의 Type 을 지정할 수 있다.

Value Type svrElementType

Defined Value

HTML
HTML(HyperText Markup Language)

CGI
CGI(Commom Gateway Inteface)

SSI
SSI(Server Side Include)

JSV
JSV(Java SerVlet)

PHP
PHP(Personal HomePage)

WEBSTD
WEB API

(164) <ws-engine> <domain> <alias> **<node-name>**

Description Alias 절이 적용되는 Node 를 지정한다. 즉, 이 경우엔 다른 Node 의 Directory 에 있는 것을 이용할 수 있다는 것이다.

Value Type len32Type

(165) <ws-engine> <domain> <alias> **<vhost-name>**

Description Alias 절이 적용되는 Virtual Host 를 제한하고 싶을 때 명시한다.

Value Type len32Type

(166) <ws-engine> <domain> **<dirindex>**

Description Client 가 요구한 index.html 과 같은 특정 파일이 존재하지 않는 경우, 전체 Directory 구조를 보여줄 수 있도록 설정한다.

Child Elements
```
(167)name
(168)option
(169)ignore*
(170)default-icon?
(171)description?
(172)header-file?
(173)tail-file?
(174)icon-ext?
```

(167) <ws-engine> <domain> <dirindex> **<name>**

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(168) <ws-engine> <domain> <dirindex> **<option>**

Description Indexing 하는 방식을 여러 가지 option 을 사용하여 설정할 수 있다.

Value Type len256Type

(169) <ws-engine> <domain> <dirindex> **<ignore>**

Description Ignore 항목에서는 Indexing 시 필요에 따라 표시하고 싶지 않은 파일들이 Index 리스트에 나타나지 않도록 해준다. Indexing 에서 제외하고 싶은 파일의 List 를 파일명을 써주거나 Wildcard(*)를 사용하여 명시한다.

Value Type len256Type

(170) <ws-engine> <domain> <dirindex> **<default-icon>**

Description Unknown File Type 에 대한 Icon 을 설정한다.

Value Type len32Type

(171) <ws-engine> <domain> <dirindex> **<description>**

Description 특정한 파일에 대한 Description 을 설정한다.

Value Type len256Type

(172) <ws-engine> <domain> <dirindex> **<header-file>**

Description Indexing 의 가장 윗부분에 집어 넣을 Header 를 지정된 파일에서 읽어

올 수 있도록 한다. 파일명은 Indexing 하고 있는 Directory 에서 상대적인 경로명으로 간주한다.

Value Type len32Type

(173) <ws-engine> <domain> <dirindex> **<tail-file>**

Description Indexing 의 가장 아래 부분에 집어 넣는 내용을 TailFile 에서 지정한 파일에서 읽어 온다. HeaderFile 과 마찬가지로 파일명은 Indexing 하고 있는 Directory 에서 상대적인 경로명으로 간주한다.

Value Type len32Type

(174) <ws-engine> <domain> <dirindex> **<icon-ext>**

Description 파일의 extension 값에 따라 그에 맞는 Icon 으로 Indexing 을 할 수 있도록 설정한다. 즉, Icon 파일의 URL 과 Mimetype 을 매칭시킨다.

Value Type len256Type

(175) <ws-engine> <domain> **<logging>**

Description Client 의 요구 내역을 기록하는 형식을 지정한다. 접근 내역과 에러 내역이 따로 저장되며 저장 형식을 지정할 수 있다.

Child Elements
```
(176)name
(177)format
(178)file-name
(179)valid-days?
(180)option?
```

(176) <ws-engine> <domain> <logging> **<name>**

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(177) <ws-engine> <domain> <logging> **<format>**

Description Log File 에 기록될 내용과 기록 방식을 설정한다. Format 이 따로 설정되지 않으면 JEUS Web Server 가 지원하는 default 로그 파일 형식이 사용된다.

Value Type len256Type

(178) <ws-engine> <domain> <logging> **<file-name>**

Description 로그를 저장할 파일의 경로명과 파일 이름을 설정한다.

Value Type len256Type

(179) <ws-engine> <domain> <logging> **<valid-days>**

Description Log 를 하나의 파일에 저장할 것인지, 하루 단위로 다른 이름으로 저장할 것인지를 결정한다.

Value Type len256Type

(180) <ws-engine> <domain> <logging> **<option>**

Description Logging 방식을 Option 을 줌으로써 바꿀 수 있다.

Value Type len256Type

(181) <ws-engine> <domain> **<access>**

Description 클라이언트의 IP 에 따라 서버의 resource 에 대한 접근 권한을 제한하거나 허가하는데 사용된다. Access 절은 DIRECTORY/URI/EXT 절에 적용될 수 있으며, 각각에서 정의한 resource 를 허용/제한하게 된다.

Child Elements
(182)name
(183)order?
(184)allow*
(185)deny*

(182) <ws-engine> <domain> <access> **<name>**

Description Access 의 이름을 설정한다.

Value Type len32Type

(183) <ws-engine> <domain> <access> **<order>**

Description Allow 와 Deny 가 적용되는 순서를 설정한다.

Value Description {"deny,allow"|"allow,deny"|"mutual-failure"}

Value Type len256Type

(184) <ws-engine> <domain> <access> **<allow>**

Description 요청이 허용되는 IP address 나 network/netmask 들을 설정한다.

Value Description {"all"|"{ipaddr|ipaddr/netmask},..."}

Value Type len1024Type

(185) <ws-engine> <domain> <access> **<deny>**

Description 요청이 거절되는 IP address 나 network/netmask 들을 설정한다.

Value Description {"all"|"{ipaddr|ipaddr/netmask},..."}

Value Type len1024Type

(186) <ws-engine> <domain> **<authent>**

Description Client 의 접근을 제한하기 위한 인증과정을 유저와 그룹 단위로 통제할 수 있도록 설정한다.

Child Elements
(187)name
(188)type
(189)user-file
(190)group-file

(187) <ws-engine> <domain> <authent> **<name>**

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(188) <ws-engine> <domain> <authent> **<type>**

Description Authorization control (인가 제어) 방식을 설정한다. Basic 방식을 지원한다.

Value Type len32Type

(189) <ws-engine> <domain> <authent> **<user-file>**

Description Authorization 을 실행하기 위한 사용자명과 암호명이 기록되는 파일을 지정한다. JEUS Web Server 는 USERFILE 관리의 편의를 돕기 위해 wsmkpw 라는 유틸리티를 제공한다. 이 유틸리티를 사용하여 사용자명과 암호화 (encrypted)된 암호명을 USERFILE 에 기록할 수 있다.

Value Type len256Type

(190) <ws-engine> <domain> <authent> **<group-file>**

Description Authorization 을 그룹단위로 실행하기 위해 사용자명과 암호명을 기록하는 파일을 따로 지정할 필요가 있을 때 사용된다. 이 파일에는 그룹의 이름과 이에 속한 구성원의 정보가 기록된다.

Value Type len256Type

(191) <ws-engine> <domain> **<ext>**

Description Client 가 요구한 파일의 확장자명에 따라 처리 담당 Process 를 지정할 수 있다. 기본적인 모든 MIMETYPE 에 대한 처리 담당 Process 가 설정되어 있으나, 필요에 따른 추가적인 설정을 할 경우 이 절에서 할 수 있다.

Child Elements
(192)name
(193)mimetype?
(194)svr-type?
(195)svr-name?
(196)charset?

(192) <ws-engine> <domain> <ext> **<name>**

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(193) <ws-engine> <domain> <ext> **<mimetype>**

Description SVRTYPE 을 설정하고자 하는 MIMETYPE 을 기록한다.

Value Type len64Type

(194) <ws-engine> <domain> <ext> **<svr-type>**

Description 앞에서 설정된 MIMETYPE 의 요구가 들어오면 이를 처리할 Server 타입을 정한다.

Value Type svrElementType

Defined Value
HTML
HTML(HyperText Markup Language)

CGI
CGI(Commom Gateway Inteface)

SSI
SSI(Server Side Include)

JSV

JSV(Java SerVlet)

PHP
PHP(Personal HomePage)

WEBSTD
WEB API

(195) <ws-engine> <domain> <ext> <svr-name>

Description 특정한 Server 가 처리하도록 설정할 경우 Server 의 이름을 설정한다.

Value Type len32Type

(196) <ws-engine> <domain> <ext> <charset>

Description 주어진 파일이름의 확장자에 대한 character set 은 명시한 charset 으로 강제화한다.

Value Type len64Type

(197) <ws-engine> <domain> <ssl>

Description Web Server 에서 사용할 SSL 의 기능을 설정하는 곳이다. 이곳에서 정의된 형태로 SSL 서비스를 하게 된다.

Child Elements

```
(198)name
(199)certificate-file
(200)certificate-key-file
(201)CA-certificate-path?
(202)CA-certificate-file?
(203)certificate-chain-file?
(204)verify-depth?
(205)verify-client?
(206)fake-basic-auth?
(207)required-ciphers?
(208)random-file?
(209)random-file-per-conn?
(210)pass-phrase-dialog?
(211)crypto-device?
```

(198) <ws-engine> <domain> <ssl> <name>

Description string 으로 사용자 임의로 정할 수 있다.

Value Type len32Type

(199) <ws-engine> <domain> <ssl> <certificate-file>

Description 이것은 PEM 방식으로 인코딩된 서버의 인증서이다. 이것은 DER 규칙으로 인코딩되어 있으며, 웹 상에서 전송하기 위해서 아스키 코드 처럼 이용된다. 만일 인증 역시 암호화 된 상태라면, 비밀 번호나 문장 등을 물어볼 것이다.

Value Type len256Type

(200) <ws-engine> <domain> <ssl> <certificate-key-file>

Description 서버에서 쓰이는 PEM 방식으로 인코딩된 인증서의 개인키를 지정한다. 만일 키가 인증서와 함께 조합되지 않았다면 이 지시자를 이용하여 키의 위치를 지정하여야 한다. 일반적으로 JEUS Web

Server 의 SSL Directory 에 가져다 두게 된다.

Value Type len256Type

`(201) <ws-engine> <domain> <ssl> <CA-certificate-path>`

Description 이 지시자를 이용하여 인증서를 저장할 디렉토리를 지정할 수 있다. 이 인증서는 받아들이기 위해 준비할 사용자의 인증서를 인증할 내용을 담고 있다. 이것은 보통 PEM 방식으로 인코딩 되어 있어야 한다.

Value Type len256Type

`(202) <ws-engine> <domain> <ssl> <CA-certificate-file>`

Description 만약 단일 CA(Certificate Agent: 인증을 대신해 주는 기업)로 부터의 사용자 인증만 받고 싶다면 CACertificatePath 지시자가 아닌, 이 지시자를 이용하여 단일 PEM 으로 인코딩된 인증 파일을 사용해야 한다.

Value Type len256Type

`(203) <ws-engine> <domain> <ssl> <certificate-chain-file>`

Description 서버 인증서(Certificate)의 인증서 연쇄(Certificate Chain)를 구성할 때 필요한, 상위 인증기관들(CAs)의 인증서(certificate) 경로를 설정한다. 단, Client 의 인증 (authentication)사용하기 위해서는 CACertificateFile 이나, CACertificatePath 에 설정해야 한다.

Value Type len256Type

`(204) <ws-engine> <domain> <ssl> <verify-depth>`

Description 실제 업무에 적용되는 경우에 인증에서 개입할 부분은 순서대로 다른 CA 에 의해서 서로를 인증하는 CA 에 관한 것이다. 이 지시자는 얼마나 깊은 레벨로 연결된 CA 들을 추적하여 인증할 것인지를 지시하는 것이다. 단 하나의 인증 CA 만 필요하다면 1 로 설정 하면 된다.

Value Type token

`(205) <ws-engine> <domain> <ssl> <verify-client>`

Description 이 지시자는 사용자에게 요청할 인증 레벨을 설정한다.

Value Type token

`(206) <ws-engine> <domain> <ssl> <fake-basic-auth>`

Description 사용자측의 인증 버전인 한줄짜리 사용자 이름을 배제한 기본 인증 과정을 통해서 마치 인증한 것처럼 보여 준다. 만약 이 지시자를 VerifyClient 지시자와 함께 설정하면 결과는 로그 파일에서 볼 수 있을 것이다. 그리고 코드는 미리 정해진 비밀 번호를 추가하여 둔다.

Value Type token

`(207) <ws-engine> <domain> <ssl> <required-ciphers>`

Description Client 가 서버와의 SSL 접속을 시작하면 정보를 암호화하는데 선호하는 Cipher 를 서버에 알리게 된다. 이 항목은 사용할 cipher 를 설정한다.

Value Type len256Type

(208) <ws-engine> <domain> <ssl> **<random-file>**

Description 이 지시자는 SSL 에서 이용할 Random Seed 를 위한 파일을 선택하는 것이다. 이 파일을 임의로 선택하면, JEUS Web Server 는 SSL 을 위한 암호를 만들 때 이 파일에서 임의의 값을 추출하여 Random Seed 를 만드는 것이다.

Value Type len256Type

(209) <ws-engine> <domain> <ssl> **<random-file-per-conn>**

Description RandomFile 을 설정할 때 실제 각 SSL 을 통한 연결을 맺을 때 몇 개의 파일을 이용하여 Random Seed 를 만들 것인지 결정하는 것이다.

Value Type len256Type

(210) <ws-engine> <domain> <ssl> **<pass-phrase-dialog>**

Description Web Server 에서 SSL 사용시에 암호화된 개인키(encrypted private key) 파일들에 대한 암호문을 얻기위한 방식을 설정한다.

Value Type len256Type

(211) <ws-engine> <domain> <ssl> **<crypto-device>**

Description SSL external Crypto Device 의 종류를 설정한다.

Value Type len256Type

(212) <ws-engine> <domain> **<errordocument>**

Description Web Server 에서 에러 문제가 발생했을 때 다음과 같은 네 가지 방법으로 대응할 수 있다. 1. 소스코드에 정의된 에러 메시지를 출력한다. 2. 사용자가 정의한 에러 메시지를 출력한다. 3. 로컬 URL 로 재전송한다. 4. 외부 URL 로 재전송한다.

Child Elements
(213)name
(214)status
(215)url?

(213) <ws-engine> <domain> <errordocument> **<name>**

Description ERRORDOCUMENT 이름을 설정한다.

Value Description String 으로 31 자 이내

Value Type len32Type

(214) <ws-engine> <domain> <errordocument> **<status>**

Description HTTP Status Code 값을 설정한다

Value Type int

(215) <ws-engine> <domain> <errordocument> **<url>**

Description DOCROOT 이하의 상대경로가 되거나 Client 가 해석할 수 있는 전체 경로 값을 설정해 주면 된다.

Value Type len256Type

(216) <ws-engine> <domain> **<expires>**

Description 클라이언트 요청에 따라 전송되는 서버응답 헤더의 정보를 설정한다. 특정 Mimetype 문서 전송시 서버응답 헤더내에 전송되는 문서의 expiry date 를 설정한다. 클라이언트 브라우저 프로그램이 같은 웹사이트에 재접속했을 때 이미 캐시에 해당 사이트의 웹 문서가 저장되어 있다면, 수정된 문서만을 받으려고 할것이다. 이때, 클라이언트의 브라우저는 웹서버에게 특정 Mimetype 과 해당 MIME 의 만료일자를 보내면서 만료일자가 경과된 문서만을 재전송해달라고 웹서버에게 요청하게 된다. 웹서버는 클라이언트의 요청헤더속에 포함된 MIME-TYPE 과 만료일자를 분석하고해당 Mimetype 문서가 현재일자보다 기간이 경과되었을때는 재전송하게 되는 것이다. 이때, 클라이언트가 웹서버에게 보내는 만료일자는 웹서버가 자신의 설정화일을 참조로 클라이언트에게 전송했던 것이다. expires 항목은 바로 이 만료일자의 설정과 관계되는 것을 정의한다.

Child Elements
```
(217)name
(218)mimetype?
(219)url?
(220)expires-time?
```

(217) <ws-engine> <domain> <expires> **<name>**

Description Expires 의 이름을 설정한다.

Value Type len32Type

(218) <ws-engine> <domain> <expires> **<mimetype>**

Description Response 의 MIME type 에 따라 expiry date 를 설정할 수 있다.

Value Type len64Type

(219) <ws-engine> <domain> <expires> **<url>**

Description Request URL 이 일치하면 해당 expiry date 를 설정할 수 있다.

Value Type len256Type

(220) <ws-engine> <domain> <expires> **<expires-time>**

Description 아래와 같이 2 가지 방법으로 Expiry Date 를 명시할 수 있다. 선택항목 없이 필수 항목만 명시한 경우, 무조건적으로 Expiry Date 가 Response Header 에 설정된다.

Value Description 1) <code><seconds> <code>: 'M': file Modification date + <seconds>로 expiry date 설정 'A': Access time + <seconds>로 expiry date 설정 <seconds>: Integer value 일반적으로 사용되는 시간 주기 (seconds): 1 hour : 3600 1 day : 86400 1 week : 604800 1 month : 2419200 2) <base> [plus] {<num> <type>}* <base>: 'access' 또는 'now': access time 'modification': file modification date [plus]: optional <num>: integer value <type>: 'years', 'months', 'weeks', 'days', 'hours', 'minutes', 'seconds', 'year', 'month', 'week', 'day', 'hour', 'minute' 또는 'second'

Value Type len256Type

Example
```
ExpiresTime = "A604800", ExpiresTime = "M3600"
ExpiresTime = "access plus 1 month", ExpiresTime
= "now plus 4 weeks", ExpiresTime = "access 1
```

```
month 15 days 2 hours"
```

(221) <ws-engine> <domain> <tcpgw>

Description Client Request 를 listener 와 request 를 처리할 server 와 관련된 정보를 설정한다. 설정된 port 를 listen 하고 있다 data 가 오면 그 data 를 그대로 설정된 서버로 분배한다. 사용 예로써는 Web Server 의 tcpgw 를 사용해서 JEUS 의 tcp-servlet 에 Round-robin 으로 data 를 분배해주는 경우가 있다. 그리고 방화벽 때문에 특정 port 만 open 이 가능한 경우 WebtoB 를 Proxy server 로 사용하는 것도 가능하다.

Child Elements

```
(222)name
(223)port+
(224)listen+
(225)server-address+
(226)timeout?
(227)connect-timeout?
(228)access-name?
(229)schedule?
```

(222) <ws-engine> <domain> <tcpgw> <name>

Description TcpGW 의 이름을 설정한다.

Value Type len32Type

(223) <ws-engine> <domain> <tcpgw> <port>

Description Web Server 에서 Client 의 Request 를 Listen 하는 Port 를 설정한다. 여러 Port 를 동시에 정의하는 것도 가능하며, 현재는 최대 100 개까지의 Port 를 동시에 설정할 수 있다. 즉, Port 의 설정을 통해서 여러 개의 Port 가 동시에 Listen 하는 것이 가능하다. 주의할 점은 Port 항목은 Listen 항목과 동시에 운영할 수 없다는 것이다.

Value Type len1024Type

(224) <ws-engine> <domain> <tcpgw> <listen>

Description Web Server 에서 Client 의 Request 를 Listen 하는 IP Address 와 Port 를 설정하게 된다. 여러개의 IP Address 와 Port 번호를 설정하는 것도 가능하며, 현재는 최대 100 개까지의 IP Address 와 Port 를 동시에 설정 하는 것이 가능하다. 여러개의 IP Address 를 가진 서버인 경우, 특정 IP Address 를 통한 요청만 받아들이기 위해서 사용할 수 있다. 주의할 점은 Port 항목은 Listen 항목과 동시에 운영할 수 없다는 것이다.

Value Type len1024Type

(225) <ws-engine> <domain> <tcpgw> <server-address>

Description Client 의 Request 를 처리 할 서버들의 IP Address 와 Port 를 설정한다. 여러개의 서버 IP Address 와 Port 를 설정하는 것도 가능하며, 현재는 최대 100 개 까지의 서버 IP Address 와 Port 를 동시에 설정 하는 것이 가능하다. 현재 설정된 서버들간의 분배규칙은 Round Robin 만 지원한다.

Value Type len1024Type

(226) <ws-engine> <domain> <tcpgw> <timeout>

Description 사용자가 Request 후, 해당 서버로부터 지정된 시간동안 응답이 없는 경우, 접속을 종료하기 위한 시간을 설정한다.

Value Description milisecond 단위

Value Type long

Default Value 300000

(227) <ws-engine> <domain> <tcpgw> **<connect-timeout>**

Description Client 의 Request 를 Web Server TCP Gateway 가 받아서, 설정된 상대 서버로 접속을 시도할 때, 일정시간 동안 접속 요구에 대한 응답이 없으면, 다른 서버로 재분배 되도록 시간을 설정한다.

Value Description milisecond 단위

Value Type long

Default Value 5000

(228) <ws-engine> <domain> <tcpgw> **<access-name>**

Description ACCESS 절의 access_name 에 설정된 규칙에 따라 사용자 접속의 허용 여부를 결정한다.

Value Type len32Type

(229) <ws-engine> <domain> <tcpgw> **<schedule>**

Description 클라이언트로 부터의 Request 을 처리할 때, 해당 Request 을 받아서 처리할 Server Processor(MinProc 개)를 지정하는 방법을 결정한다.

Value Type len256Type

Default Value RR

(230) <ws-engine> <domain> **<nlive-inq>**

Description JEUS Web Server 가 여러 Node 로 구성되었을 경우 Node 간 감시 Interval(초 단위) 시간을 정하는 항목이다. NLiveInq 시간이 짧으면 양 Node 간에 빈번한 감시로 시스템의 이상을 빨리 발견할 수 있으나 많은 통신량으로 인해 통신 부하가 발생할 수 있다. 반대로 NLiveInq 시간이 너무 길면 통신량은 적으나 Node 의 이상 상태를 즉시 감지 할 수 없다. 따라서 시스템 성능을 최적화하기 위하여 네트워크 부하량, 업무의 중요도 등을 고려하여야 한다.

Value Description Second 단위

Value Type int

Default Value 30

F.4 WSMain.xml 파일 예제

<<WSMain.xml>>

```
<ws-engine xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
   <domain>
      <name>jeuservice</name>
      <node>
         <name>tmax</name>
         <webtob-dir>
            D:/JeusSrc/jeus5/webserver
         </webtob-dir>
         <shmkey>45400</shmkey>
         <docroot>
            D:/jeus50cvs/jeus5/webserver/docs
         </docroot>
         <jsv-port>9800</jsv-port>
         <port>8088</port>
         <logging>log1</logging>
         <logging>log2</logging>
         <error-log>log2</error-log>
      </node>
      <svrgroup>
         <name>htmlg</name>
         <svr-type>HTML</svr-type>
         <node-name>tmax</node-name>
      </svrgroup>
      <svrgroup>
         <name>jsvg</name>
         <svr-type>JSV</svr-type>
         <node-name>tmax</node-name>
      </svrgroup>
      <server>
         <name>html</name>
         <svg-name>htmlg</svg-name>
         <min-proc>2</min-proc>
         <max-proc>10</max-proc>
      </server>
      <server>
         <name>MyGroup</name>
```

```
        <svg-name>jsvg</svg-name>
        <min-proc>4</min-proc>
        <max-proc>10</max-proc>
    </server>
    <uri>
        <name>uri</name>
        <uri-def>/examples/</uri-def>
        <svr-type>JSV</svr-type>
        <svr-name>MyGroup</svr-name>
    </uri>
    <logging>
        <name>log1</name>
        <format>DEFAULT</format>
        <file-name>
            D:/jeus50cvs/jeus5/webserver/log/access.log
        </file-name>
        <option>sync</option>
    </logging>
    <logging>
        <name>log2</name>
        <format>ERROR</format>
        <file-name>
            D:/jeus50cvs/jeus5/webserver/log/ error.log
        </file-name>
        <option>sync</option>
    </logging>
    <ext>
        <name>htm</name>
        <mimetype>text/html</mimetype>
        <svr-type>HTML</svr-type>
    </ext>
    <ext>
        <name>jsp</name>
        <mimetype>application/jsp</mimetype>
        <svr-type>JSV</svr-type>
    </ext>
  </domain>
</ws-engine>
```

색인

U

V

W