관리자 안내서

WebtoB 5 Fix#4

TMAXSOFT

저작권 공지

제한된 권리

상표 공지

오픈소스 소프트웨어 공지

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다. : OpenSSL, ZLIB, PCRE, APACHE1.0, APACHE1.1, APACHE2.0, JSON-C, BSD, RSA, PHP, Paul Hsieh’s hash

관련 상세한 정보는 제품의 다음의 디렉터리에 기재된 사항을 참고해 주십시오. : ${INSTALL_PATH}/lib/licenses

안내서 이력

제품 버전	안내서 버전	발행일	비고
WebtoB 5 Fix#4	2.1.5	2020-11-06	-

목차

1. WebtoB 소개

본 장에서는 WebtoB의 구조와 동작, 특징에 대해서 알아보고, WebtoB를 사용하기 위해 알아야 하는 관리 툴과 환경변수에 대해서 설명한다.

1.1. 구조와 동작

WebtoB는 기존 웹 서버와는 다른 구조를 가지고 있다. 기존 웹 서버들은 대부분 NCSA사의 httpd의 구조를 가지고 있다. 이는 사용자가 많지 않던 환경에서 사용되던 것으로 사용자의 증가에 유연하게 대처하지 못하는 단점을 지니고 있다.

WebtoB에서는 이런 문제를 해결하기 위해 최우선적으로 고려한 것이 사용자가 계속 증가하는 경우에 대한 대처 방식이었다.

기존의 웹 서버들은 사용자의 Request가 들어오면 1:1 방식으로 연결을 맺는 구조로 되어 있다. 따라서 사용자가 증가하는 경우 결국 서버 프로세스나 스레드가 그 수만큼 동시에 늘어나야 서비스를 할 수 있다. 그러나 이러한 방식은 사용자가 증가하면 할수록 서버에 걸리는 부하도 따라서 증가하기 때문에 많은 문제점을 야기한다.

물론, 사용자가 증가하면 서버에 발생하는 부하가 적을 수는 없다. 더 많은 메모리와 CPU Overhead를 감수해야 하는 것은 당연한 일이다. 그러나 사용자 증가로 인해 서버에 발생하는 부하가 비례적으로 증가하여 더 많은 하드웨어를 추가로 구입해야 한다면, 이는 적절한 해답이 될 수 없다. 즉, 사용자가 증가해도 서버 자체에서 적절히 대응하여 일정 수준까지는 무리 없이 서비스를 가능하게 하는 구조가 되어야 무한하게 확장하는 웹에 대응할 수 있는 것이다.

이어지는 절에서 기존의 웹 서버와 WebtoB와의 구조에 대해 설명하고 WebtoB의 구조가 웹 환경에 대응할 수 있는 이유를 알아본다.

1.1.1. 기존 웹 서버와의 구조 비교

초기의 웹 서버는 httpd의 구조를 그대로 계승했기 때문에 가장 일반적인 Process per Request 형태를 취하고 있다. 즉, 사용자 Request마다 프로세스가 발생해서 처리하는 구조이다. 따라서 앞에서 언급한 대로 사용자가 증가하면 프로세스도 같이 비례하여 증가하게 되므로, 사용자 수가 일정 수 이상이 되면 서비스에 많은 무리가 생긴다. 이는 각 시스템마다 프로세스의 수가 한정되어 있고, 프로세스가 증가하면 너무 많은 메모리를 차지하는 등의 Overhead도 함께 증가하기 때문이다.

이로 인하여 새로운 상용 웹 서버들이 등장하면서 급속도로 확장하고 있는 웹 환경에 대처하기 위하여 새로운 구조를 채택하였다. 가장 일반적인 것이 Multi Thread 기술을 이용한 것이다. 이는 프로세스보다 Overhead가 적은 Thread를 사용하여 서비스하는 것으로 기존 프로세스 방식에 비해 월등히 좋은 성능을 나타냈다. Thread는 프로세스 내부에서 여러 개를 동시에 이용할 수 있기 때문에 Overhead도 적고, 차지하는 메모리도 프로세스 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 DeadLock이 발생하면 자체 Thread가 서비스를 수행하지 못함은 물론, 자칫하면 전체 시스템이 다운되어 서비스 불능 상태가 될 수 있다. 이는 성능뿐 아니라 안정성도 중요한 요소가 되는 웹 서버에 치명적인 결함이 될 수 있다.

위에서 언급한 2가지 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 서비스가 되지 않기 때문이다.

TmaxSoft에서는 WebtoB를 개발하면서 성능과 안정성을 모두 충족시키기 위하여 많은 구조를 검토하고 연구하였다. WebtoB에서는 안정성을 위하여 프로세스 구조를 채택함과 동시에 성능을 위해 스레드 구조를 접목하였다. 안정 및 성능을 위해 각 프로세스를 독립적으로 만들지 않고 각각의 특별한 기능을 하는 프로세스를 두어 역할을 분담하고, 특정 프로세스의 경우 특별한 기능을 하는 스레드를 두어 역할 을 분담하였다. 또한 각 프로세스마다 만약의 경우를 대비하여 여분의 보조 프로세스를 두는 것을 가능하게 하여 안정성에 치중하였다.

1.1.2. WebtoB 프로세스 및 스레드

다음은 WebtoB의 주요 프로세스와 스레드에 대한 설명이다.

- **WSM(WebtoB System Manager)**

  전체적인 WebtoB 시스템의 운용 프로세스로써 시스템의 운영 정보를 관리하고, HTL/HTH 프로세스 및 모든 서버 프로세스들을 관리하는 프로세스이다. WebtoB 시스템을 기동할 때 WSM은 가장 먼저 메모리에 로드되고 시스템이 종료될 때에는 가장 나중에 종료된다.

- **HTL(HTTP Listener)**

  HTL은 클라이언트와 WebtoB 간의 연결을 관리하는 Listener 프로세스이다. 클라이언트가 처음 WebtoB에 접속할 때에는 HTL과 연결을 맺어 통신이 이루어진다. 하지만 서비스 요청이 있을 경우 내부적으로 HTH와 연결이 되어 모든 서비스 처리가 이루어진다.

- **HTH(HTTP Handler)**

  클라이언트 핸들러라고도 하며 실질적으로 클라이언트와 서버의 업무 처리 프로세스 사이를 중계하는 프로세스이다. HTH는 서버 프로세스들과의 통신을 통해 모든 실제적인 데이터의 흐름을 관리한다. 즉 클라이언트의 서비스 요청을 받아 그에 해당하는 업무를 처리하며 그 결과를 수신하여 다시 클라이언트에게 되돌려준다. 최대한 락을 잡지 않는 고성능 아키텍처의 스레드 모델로 구성되어 있으며 각 스레스별로 아래와 같은 역할을 담당한다.

스레드 타입	설명
ACCESSLOG	HTH1개당 1개만 존재할 수 있는 스레드이며 HTTP(S) 요청 처리 후 accesslog를 기록한다.
WORKER	HTML 요청을 처리하거나 SSL, Compression 등을 처리한다.
SENDFILE	HTML 요청을 블록킹으로 클라이언트에게 전달한다.

- **PHPS(PHP Server)**

  PHP 요청을 처리하는 PHP 서버 프로세스이다.

- **CGIS(CGI Server)**

  CGI 요청을 처리하는 CGI 서버 프로세스이다.

- **SSIS(SSI Server)**

SSI 요청을 처리하는 SSI 서버 프로세스이다.

1.2. 특징

WebtoB의 대표적인 특징을 살펴보면 다음과 같다.

1.2.1. 캐싱(Caching)

일반적으로 많은 웹 서버들이 캐싱 기능을 제공하고 있다. 그러나 이들은 대부분 디스크에 대한 캐싱으로서 필요한 데이터들을 다른 머신에서 자신의 머신으로 가져와서 디스크에 저장해 두고, 사용자가 Request를 하면 이를 보내주는 방법이다. 주로 Proxy라는 개념으로 많이 이용되는 다른 웹 서버도 역시 이 기능을 가지고 있으며, 이는 성능 향상에 많은 도움을 준다.

WebtoB에서의 캐싱은 사용자가 Request를 보내면 WebtoB가 자주 이용되는 리소스들을 선별하여 이를 메모리에 상주시켜 놓는 시스템이다.

현재 웹 서비스를 하고 있는 많은 회사들은 상당히 고가의 장비를 이용한다. 이들은 대부분 막강한 능력의 CPU를 여러 개 탑재하고 있으며, 또한 수십 GB에 이르는 엄청난 양의 메모리를 가지고 있다. 그러나 실제 서비스를 시행하게 되면 CPU의 처리 능력에 의해 성능의 제약이 오고, 메모리를 효율적으로 이용하지 못하는 경우가 많게 된다. 즉, 대부분의 경우 메모리의 모든 영역을 쓰지는 못하고, CPU의 처리 능력에 의해서 메모리의 70~80% 정도만 이용하는 실정이다.

이에 WebtoB에서 제공하는 메모리 캐싱 기능은 이 여분의 메모리에 자주 이용되는 리소스들을 미리 상주시켜 두어 성능 향상을 돕는다. 보통 디스크에 비해서 메모리가 약 수십 배 가량 처리 속도가 빠르기 때문에 이것을 적절히 활용하면 대단히 큰 성능 향상을 가져온다. 또한 저장 용량을 임의로 조정할 수 있어 메모리가 부족한 경우 이를 줄일 수 있으며, 만약 여분의 메모리가 충분하다면 이 크기를 더 늘일 수 있다.

실제 연구 결과를 보면 사용자 Request의 대부분이 특정한 리소스에 몰려 있게 된다. 따라서 작은 용량의 메모리 캐싱이라 할지라도 이들을 이용하게 되면 엄청난 성능 향상을 보일 수 있게 된다.

1.2.2. WBAPI

WebtoB는 WBAPI라는 내부 함수를 제공한다. 이는 WebtoB에서만 제공하는 것으로 다양한 용도로 적용될 수 있다. 우선 기존 CGI 프로그램들을 변환하는 데 이용할 수 있다. 기존 CGI 프로그램은 상당히 비효율적으로 설계되어 있고 구현은 간단하지만 성능 면에서 워낙 문제가 많기 때문에 사용자가 많은 곳에 적용하기는 힘들었다. 기존 사이트 중 CGI 프로그램으로 애플리케이션을 개발하고 서비스를 하는 중 갑자기 사용자가 늘어나는 경우에는 본래 이용하던 CGI 프로그램으로는 감당하기 힘들게 되는 경우가 많이 있다. 이 경우 현재까지의 해결 방법은 CGI 프로그램과 같은 기능을 하는 다른 애플리케이션을 완전히 새로 개발하거나 하드웨어를 무한정 늘리는 것 외에는 다른 방법이 없었다.

WebtoB에서는 WBAPI를 제공하여 기존 CGI 프로그램을 WebtoB의 서비스 형태로 변환하는 것을 가능하게 한다. 이는 기존 CGI 프로그램의 단점을 모두 개선한 형태로 현재 많이 이용되는 Servlet이나 기타 다른 스크립트 언어 등과 같은 성능을 제공한다. 이는 또한 WebtoB에서 TP-Monitor의 서비스 루틴을 호출하는 데 이용되기도 한다.

1.2.3. TP-Monitor Tmax 서비스 호출

브라우저에서 WebtoB로 Tmax 서비스를 호출하게 되면 WebtoB에서 정해진 형태의 Request를 보내게 되는데, 일반적인 형태는 CGI Request와 거의 같다고 보면 된다. 다만 사전에 WebtoB 내에서 특정 Request가 들어오면

Tmax 서비스로 인식한다는 설정만 하면 된다.

예를 들면 WebtoB 내의 "/tmax" 디렉터리에 Tmax 서비스가 존재한다고 가정하고 여기에 service1이라는 서비스가 존재한다면 브라우저에서는 다음과 같은 형태의 Request를 보내면 된다.

```
http://www.tmax.co.kr/tmax/service1
```

1.2.4. Extension 관리

일반적으로 웹 서버가 기본적으로 제공하는 MIME-Type은 정해져 있다. 이들은 거의 표준화되어서 특정 데이터나 프로그램들이 전송되어져야 할 필요가 있을 때 이에 대한 정의가 웹 서버 내에서 이루어져야 한다. 따라서 사용자가 새로운 형태의 데이터형을 생성해서 사용하고 싶다면 새로운 MIME-Type을 만들어야 한다. WebtoB에서는 이러한 데이터형을 관리자가 정의하여 제공할 수 있다.

또한 자신이 원하는 데이터의 Extension(확장자)를 임의로 설정이 가능하여 사용자 자신만의 Extension을 생성할 수 있다.

1.2.5. Log 관련

다른 웹 서버에서 제공하는 대부분의 Log Format을 만들 수 있다. 또한 사용자가 원하는 형식으로 조정이 가능하다.

1.3. 관리 툴

WebtoB는 엔진 프로세스 및 서버 프로세스들을 관리하기 위해서 다음과 같은 툴을 제공한다.

관리 툴	설명
wsadmin	WebtoB 시스템 전체적인 관리를 위해서 사용되는 툴로서 시스템 정보 및 관리자의 작업들을 수행할 수 있다.
wscfl	텍스트 기반의 환경 파일을 컴파일해서 바이너리 파일로 변환한다.
wsuncfl	컴파일된 바이너리 파일을 텍스트 기반의 환경 파일로 변환한다.
wsmkpw	사용자 인증에 관련된 정보들을 암호화해서 특정 파일을 생성한다.
wsmkppd	*SSL.PassPhraseDialog 설정 중 "file:<passphrase file path>"로 설정할 때 사용할 passphrase 암호를 저장하고 있는 파일을 생성한다.

1.4. 환경변수

다음은 WebtoB에서 설정하는 환경변수에 대한 설명이다.

환경변수	설명
WEBTOBDIR	WebtoB가 설치된 디렉터리 정보를 설정한다. (필수 항목)
WEBTOB_LICENSE	라이선스 파일명을 변경하는 경우 설정한다. (기본값: license.dat)
WEBTOB_RAC_PORT	**wsracd**가 사용하는 포트 번호를 변경하는 경우 설정한다. (기본값: 3333)

환경변수	설명
WEBTOB_PREFER_IPV6	IPv6 주소를 사용하고자 할 경우 'Y' 혹은 '1'로 설정한다.

추가적으로 다음의 환경변수에 WebtoB의 Shared library가 위치한 경로인 "${WEBTOBDIR}/lib"를 시스템 라이브러리 경로(System library path)로 추가해야 한다.

환경변수	설명
LD_LIBRARY_PATH	Linux 및 UNIX 계열의 일반적인 시스템 라이브러리 경로이다.
LD_LIBRARY_PATH_64	Solaris(SunOS)에서 64-bit WebtoB를 사용할 때 적용되는 시스템 라이브러리 경로이다.
SHLIB_PATH	HP-UX에서 32-bit WebtoB를 사용할 때 적용되는 시스템 라이브러리 경로이다.

1.5. 디렉터리 구성

WebtoB 시스템이 사용하는 디렉터리 구성은 다음과 같다.

```
WebtoB
  |-- bin
  |-- lib
  |-- usrinc
  |-- ap
  |-- config
  |-- path
  |-- log
  |-- docs
  |-- license
```

bin

실행 파일들이 위치한다. (wsm, wscfl, wsracd, wsgst, wsboot, wsdown, etc)

lib

라이브러리 파일이 위치한다.

usrinc

API 함수들의 Header 파일이 위치한다.

ap

사용자 프로그램이 위치한다.

config

WebtoB 환경 파일이 위치한다.

path

프로세스 내부 통신을 위한 Named Pipe가 생성된다.

log

로그 파일들이 위치한다.

docs

WebtoB에서 기본적으로 등록되는 HTML 문서가 위치한다.

license

WebtoB 라이선스 파일이 위치한다.

1.6. 라이선스별 지원 기능

WebtoB는 라이선스 타입별로 다음과 같은 기능을 제공한다.

라이선스 타입	설명
TRIAL	Installer로 설치한 WebtoB에는 기본적으로 트라이얼 라이선스(Trial License)가 내장되어 있다. HTH는 1개로 제한되고, Max user 수가 5개로 제한된다.
BASE	JEUS의 내장 WebtoB로 사용되는 경우이다. HTH가 1개로 제한된다. UNIX/Linux 환경에서는 도메인 소켓(named pipe)을 통해서만 JEUS와 연동할 수 있기 때문에 JSVPort를 사용하지 않는다.
STANDARD	BASE와 달리 HTH가 1개 이상 지원되는 일반적인 경우이다. JEUS와도 자유롭게 연동할 수 있으며 WebtoB가 제공하는 대부분의 기능을 사용할 수 있다.
ENTERPRISE	STANDARD에서 제공하는 모든 기능 이외에 아래와 같은 추가적인 기능을 사용할 수 있다. ◦ 내장된 JEUS의 JSP/Servlet Engine을 사용할 수 있다. 이 경우 내장 JEUS의 1개 서버(DAS)를 사용할 수 있다. ◦ Reverse Proxy를 이용한 타 WAS를 연동할 때 다중 구성을 위한 Reverse Proxy Group 기능을 사용할 수 있다. ◦ WebDAV를 위한 HTTP Method(MKCOL, COPY, MOVE, PORPFIND)를 사용할 수 있다. ◦ 필터(Filter) 처리를 위한 FILTERS 프로세스를 사용할 수 있다. 특히 CA사의 SSO 연동을 위한 SiteMinder Filter(wbSmISAPI)를 사용할 수 있다. ◦ WebAdmin을 사용할 수 있다.

2. Quick Start

본 장에서는 간단한 WebtoB의 사용 방법에 대해서 설명한다.

WebtoB가 설치된 장비의 hostname은 "mynode"라고 가정한다. Installer로 WebtoB를 정상적으로 설치한 경우 WEBTOBDIR과 PATH 등 환경변수가 WebtoB의 환경 파일에 설정되어 있는지 확인한다.

WebtoB의 설치에 대한 자세한 설명은 WebtoB 설치 안내서를 참고한다.

1. WebtoB의 환경 파일 컴파일 툴인 **wscfl**을 사용해서 http.m 파일을 컴파일한다. 자세한 설명은 환경설정을 참고한다.

```
$ wscfl -i http.m
```

<$WEBTOBDIR\config\http.m>

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 54000,
  Docroot="docs/",
  Port = "8080",
  HTH = 1,
  Logging = "access_log",
  ErrorLog = "error_log",
  SysLog = "system_log"

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
  SvrType = CGI
ssig
  SvrType = SSI

*SERVER
cgis
  SvgName = cgig,
  MinProc = 2,
  MaxProc = 10,
  ASQCount = 100
ssis
  SvgName = ssig,
  MinProc = 2,
  MaxProc = 10,
  ASQCount = 100

*URI
cgi-bin
```

```
  URI = "/cgi-bin/",
  SvrType = CGI

*ALIAS
cgi-bin
  URI = "/cgi-bin/",
  Realpath = "cgi-bin/"

*LOGGING
access_log
  Format = "default",
  Filename = "log/access.log",
  ArchiveFilename = "log/access_%Y%M%D%.log",
  Option = "sync"
error_log
  Format = "",
  Filename = "log/error.log",
  ArchiveFilename = "log/error_%Y%M%D%.log",
  Option = "sync"
system_log
  Format = "",
  Filename = "log/webtob.log",
  ArchiveFilename = "log/webtob_%Y%M%D%.log",
  Option = "sync"
```

2. "${WEBTOBDIR}\config\" 디렉터리에 바이너리 설정 파일인 **wsconfig**가 생성되었는지 확인한다.
3. WebtoB를 기동하기 위해 명령 프롬프트에서 **wsboot**라고 입력한다.
4. 브라우저를 열고, 다음과 같이 URL을 입력한다.

   ```
   http://[IP Address]:[8080 또는 사용자가 지정한 PORT 번호]/
   ```

   정상적으로 WebtoB가 기동된 경우 다음과 같은 화면이 표시된다.

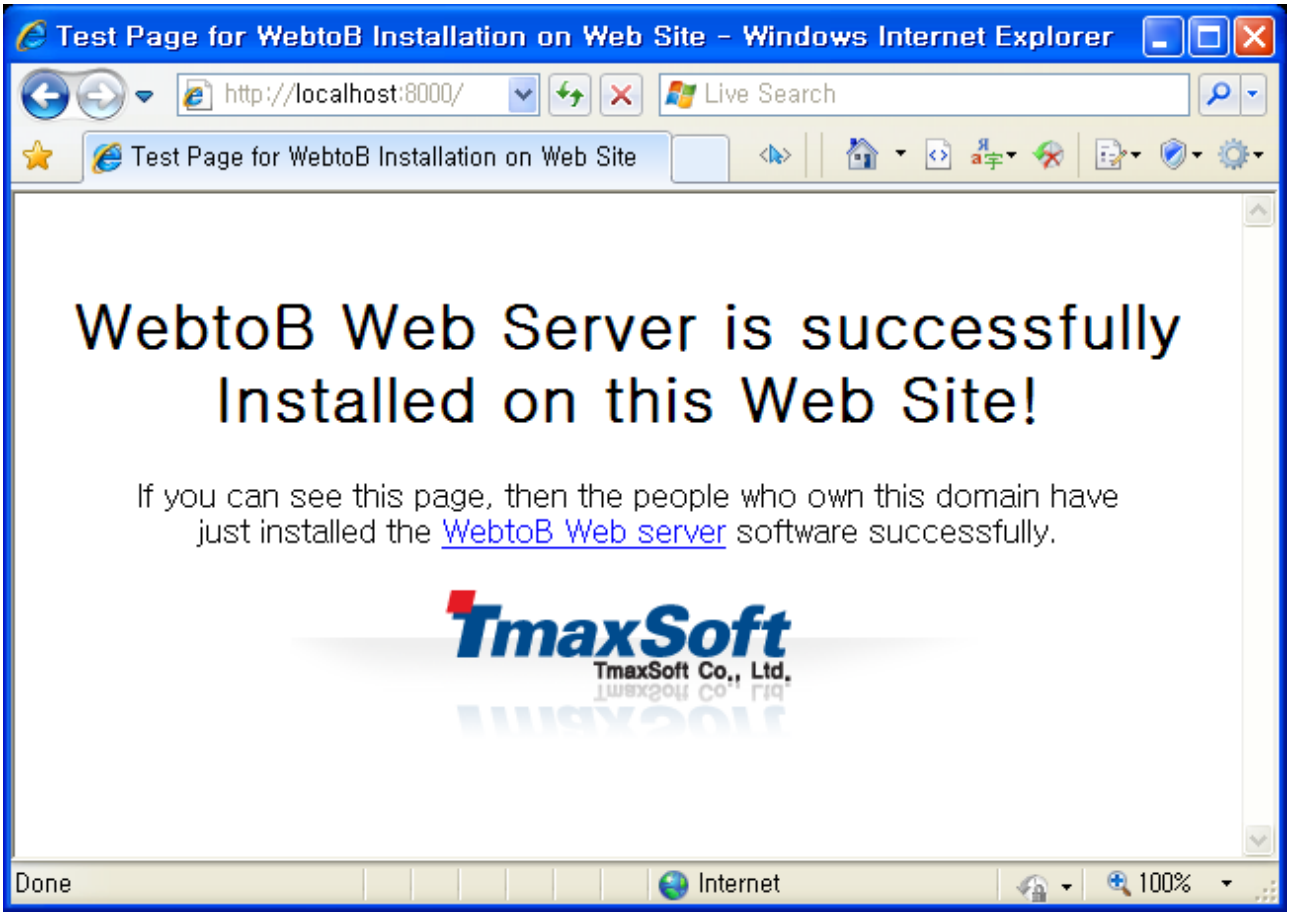


그림 1. WebtoB 테스트 페이지

5. WebtoB 관리 툴을 띄우기 위해 **wsadmin**을 입력한다.

   wsadmin에서 'help'나 'h'를 입력하면, WebtoB를 모니터링하거나 동작을 제어할 수 있는 명령어의 목록이 조회된다. 'quit'를 입력하여, WebtoB 관리 툴을 종료한다.

   > wsadmin의 사용법에 대한 자세한 내용은 wsadmin 콘솔 관리자 프로그램을 참고한다.

6. **wsdown** 명령으로 WebtoB를 종료한다.

3. 환경설정

본 장에서는 환경설정의 구조와 각 절의 설정 항목과 예제에 대해서 설명한다.

3.1. 개요

WebtoB 환경설정은 하나의 WebtoB 도메인(Domain) 및 노드(Node)에 대한 설정과 웹 서버의 동작에 대한 설정을 한다.

3.1.1. 환경설정 파일 구조

WebtoB의 환경설정 파일은 크게 절을 정의하는 부분과 설정할 이름과 해당 절에 설정할 수 있는 항목으로 구성된다.

```
# 주석
*절
이름
    # 주석
    항목 = 설정값, # 주석
    ...,
    항목 = 설정값
```

절을 시작한 이후 정의할 이름을 설정하고, 각 절에서 설정할 수 있는 각각의 항목을 설정한다.

다음은 WebtoB에서 설정해야 하는 절에 대한 설명이다.

절	설명
DOMAIN 절	WebtoB 도메인을 설정한다.
NODE 절	WebtoB 노드에 대한 구체적인 환경에 대해 설정한다.
VHOST 절	Virtual Hosting이 필요한 경우 VHOST 절에 관련 환경을 설정한다.
HTH_THREAD 절	HTH 프로세스 내 Thread를 설정한다.
SVRGROUP 절	WebtoB를 통해 응용 서버 프로세스를 접근하는 경우 서버 프로세스의 논리적인 연관성에 따라 이들을 그 룹으로 관리할 경우 설정한다.
SERVER 절	요청을 처리할 프로세스와 관련된 설정을 한다.
SERVICE 절	WebtoB를 통해 비즈니스 로직을 바로 수행할 경우 설정한다.
DIRECTORY 절	노드 내의 특정 디렉터리의 속성을 설정한다.
URI 절	클라이언트 요구의 URI(Uniform Resource Identifier) 값에 따라 이를 처리하는 서비스를 구분할 수 있도록 한다.
ALIAS 절	실제 서버의 물리적 디렉터리 경로와 URI를 Alias시키도록 설정한다.
DIRINDEX 절	인덱싱하는 방식과 Icon 등을 지정할 수 있다.
LOGGING 절	클라이언트의 요구 내역을 기록하는 형식을 지정한다.
ACCESS 절	클라이언트에서 접속을 시도할 때 IP 주소, network/netmask, Header 정보들을 기준으로 요청의 허용/제 한을 설정한다.

절	설명
AUTHENT 절	클라이언트의 접근을 제한하기 위한 인증 과정을 사용자와 그룹 단위로 통제할 수 있도록 설정한다.
EXT 절	클라이언트가 요구한 파일의 확장자명에 따라 처리 담당 프로세스를 설정한다.
SSL 절	WebtoB에서 사용할 SSL의 기능을 설정한다.
PROXY_SSL 절	WebtoB가 Proxy 역할을 할 때 사용할 SSL의 기능을 설정한다.
ERRORDOCUMENT 절	에러 문제가 발생했을 때 대응방법을 설정한다.
EXPIRES 절	클라이언트 요청에 따라 전송되는 서버 Response Header의 정보를 설정한다.
TCPGW 절	특정 Port나 IP 주소로 들어오는 TCP 연결을 다른 서버로 전송하는 중계(proxy)하는 역할을 한다.
REVERSE_PROXY_GROUP 절	REVERSE_PROXY 절 설정을 그룹으로 관리할 필요가 있을 때 사용한다.
REVERSE_PROXY 절	HTTP Proxy 일종으로 외부에서 접근 불가능한 내부 서버로 요청을 전달할 때 설정한다.
LOGLEVEL 절	시스템 로그 메시지 출력을 제어하는 로그 레벨을 설정한다.
HEADERS 절	사용자 요청 및 응답에 특정 HTTP Header를 추가하는 경우 설정한다.
PRECEDING_COMMAND 절	WebtoB를 기동할 때 WebtoB의 각 프로세스별 CPU Affinity를 지정하는 등의 선행 명령어를 실행하도록 설정한다.
FILTER 절	Filter 모듈을 사용할 FILTER 절을 정의하여, NODE 절이나 VHOST 절 또는 SVRGROUP 절에 Filter 항목 을 설정한다.
USERLOGFORMAT 절	공통으로 사용될 로그 포맷을 설정한다.
DISK_CACHE 절	디스크 캐싱을 사용할 경우 설정한다.
LOG_HANDLER 절	WebtoB에서 accesslog를 리모트 서버에 남기는 경우 설정한다.
IPGROUP 절	설정 값이 여러 개의 IP인 경우 이를 그룹으로 관리할 필요가 있을 때 사용한다.

DOMAIN, NODE, HTH_THREAD 절은 반드시 설정해야 한다.

절은 다음의 규칙을 준수해서 설정해야 한다.

- 절은 애스터리스크(*) 이후 절의 이름을 설정하여 시작한다.
- 절의 시작은 줄의 첫 번째 칸에 애스터리스크(*)로 시작하며, 절의 이름은 대문자를 사용한다.
- 절을 시작한 줄에는 다른 설정이 있어서는 안 되며, 적어도 하나 이상 해당 절과 관련된 설정을 해야 한다.
- 절을 정의한 이후부터 다른 절을 정의하기 전이나 설정의 끝까지는 해당 절에 관련된 설정으로 간주한다.
- 절의 순서는 고정되어 있지 않기 때문에 어느 절이 먼저 정의되어도 상관없다.
- 절의 정의는 내용별로 나누어 한 번 이상 나타날 수 있다. 즉, 하나의 절을 여러 번으로 나누어 정의할 수 있다.

절 이름

절에는 해당 절을 구분하기 위해 절 이름을 지정해야 한다. 절 이름은 다음의 규칙을 참고해서 설정해야 한다.

- 절의 이름은 줄의 첫 번째 칸에서 시작해야 한다.
- 절의 이름은 알파벳과 숫자, 하이픈(-), 마침표(.), 언더라인(_)을 사용할 수 있다. 단, SERVICE 절의 이름은 알파벳과 숫자, 언더라인(_)만 사용할 수 있다.
- 같은 절에서 절의 이름은 동일한 값을 사용할 수 없다. 즉, 동일한 이름으로 반복 정의해서는 안 된다.

항목 설정

절의 이름을 설정한 이후에 "항목이름 = 설정값"의 형태로 설정한다.

```
*절
이름
    항목1 = 설정값,
    항목2 = 설정값,
    ...,
    항목n = 설정값
```

항목은 다음의 규칙을 참고해서 설정한다.

- 설정할 항목이 여러 개일 경우 콤마(,)로 구분하여 설정한다.

항목 설정이 콤마(,)로 끝나는 경우 다음 줄에 다른 항목 설정이 있다는 것을 의미한다. 따라서 마지막 항목 설정은 콤마(,)로 끝나서는 안 된다.

- 항목명은 대소문자를 구분하지 않는다.
- 항목 설정은 줄의 첫 번째 칸에서 시작할 수 없다.
- 반드시 하나 이상의 공백 이후에 설정해야 한다.
- 여러 항목을 같은 줄에 설정할 수도 있다.
- 항목 설정은 기본적으로 같은 항목에 대해 하나만 설정하지만, 항목에 따라서는 반복적으로 설정할 수도 있다.
- 항목은 각 절에 반드시 설정해야 하는 필수 항목이 있을 수 있으며, 나머지 항목은 필요한 경우에만 설정한다.

항목의 설정값은 항목에 따라 Numeric, String, Literal, Boolean 형식으로 설정하며, 각각 다음과 같이 설정한다.

종류	설명
Numeric	숫자를 설정한다. 0700처럼 0으로 시작하는 경우 8진수로, 0xff처럼 0x로 시작하는 경우 16진수로 처리한다.
String	문자열을 설정한다.
Literal	문자열을 설정한다. 항상 따옴표(" ")로 묶여 있어야 한다.
Boolean	Y/N 또는 YES/NO로 설정한다.

주석

해시 기호(#) 이후부터 그 줄의 마지막까지는 주석으로 간주한다.

설정 예제

```
*DOMAIN
webtob

*NODE
mynode
    WebtoBDir = "$WEBTOBDIR",
    SHMKEY = 54000
    # Port = "80" is default

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
    SvrType = CGI

*SERVER
cgis
    SvgName = cgig
    # MinProc = 1, MaxProc = 1 is default
```

3.1.2. 환경설정 파일 컴파일

WebtoB를 실행하기 전에 테스트로 작성된 설정 파일은 **wscfl**을 사용해서 실제 WebtoB가 인식할 수 있는 바이너리 형태로 전환(컴파일)해야 한다.

텍스트 형식의 설정 파일은 기본적으로 "http.m"을 사용하며, 바이너리 형태의 설정 파일은 "wsconfig"를 사용한다. 컴파일 과정에서 wscfl은 텍스트 형식으로 작성된 설정 파일의 문법적 오류나 시스템 설정 오류를 검증한다.

다음은 텍스트 설정 파일 "$WEBTOBDIR/config/http.m"을 컴파일하는 예제이다.

```
wscfl -i http.m
```

바이너리 형태인 설정 파일의 기본 파일명인 "wsconfig"를 다른 이름으로 사용하는 경우 WEBTOB_CONFIG 환경변수를 설정하거나, wscfl 및 wsadmin, wsboot, wsdown 프로그램을 실행할 때 옵션으로 파일명을 지정할 수 있다.

3.1.3. 설정 항목 설명 규약

다음 절에서 각 절의 설정 항목을 설명할 것이다. 설명하는 각 절의 설정 항목(회색박스)의 내용을 이해하기 위해서 다음의 규약을 알고 있어야 한다.

- 이름 앞에 해시 기호(#)로 시작하는 항목은 선택 항목임을 나타낸다.

- String 및 Literal 형식의 경우 대괄호([ ])안의 값은 내부적으로 사용하는 버퍼의 크기를 의미한다. String[n] 혹은 Literal[n]의 경우 설정할 수 있는 문자열의 길이가 (n -1)로 제한된다.
- 기본값이 있는 설정의 경우 해시 기호(#) 이후에 표시한다.
- Numeric 형식의 경우 괄호를 사용하여 설정 가능한 범위를 표시한다.
- 최댓값 제한이 없을 경우 생략하며, INT_MAX(2147483647)까지만 사용할 수 있다.
- "$ENV"는 환경변수를 참조할 수 있다는 의미이다.
- "R.PATH"는 상대 경로로 설정하는 경우 $WEBTOBDIR로부터의 상대 경로로 사용한다는 의미이다.
- "LIST"는 콤마(,)를 사용하여 여러 설정을 동시에 할 수 있다는 의미이며, "LIST[n]"은 최대 n개로 동시에 설정가능한 수가 제한됨을 의미한다. [n]이 없을 경우 전체 길이가 버퍼의 크기보다 작게만 설정하면 된다.
- "PM.LIST"는 설정 앞에 플러스 기호(+) 혹은 마이너스 기호(-)를 사용하여 설정할 수 있음을 의미한다. 보통 플러스 기호의 경우 해당 기능을 사용한다는 의미이며, 생략할 수 있다. 마이너스 기호는 해당 기능을 사용하지 않도록 설정하라는 의미이다.
- "MULTI"는 같은 항목을 여러 번 설정할 수 있다는 의미이며, "MULTI[n]"은 최대 n개로 동시에 설정 가능한 수가 제한됨을 의미한다.
- "LIST"와 "MULTI"가 같이 있을 경우 동시에 여러 번 설정도 가능하고, 같은 항목을 여러 번 설정하는것도 가능하다는 의미이다. 이 경우 최대 설정 가능한 수는 LIST[n]의 값으로 제한된다.

3.2. DOMAIN 절

DOMAIN 절은 WebtoB 도메인을 설정한다. DOMAIN 절은 반드시 설정해야 하며, 하나만 설정한다.

3.2.1. 설정 항목

다음은 DOMAIN 절의 환경설정 형식이다.

```
*DOMAIN
name    # String[32]
        #DomainID = Numeric,             # 0 (0-255)
        #MaxSvc = Numeric,               # 512 (1-0xffff)
        #NHthChkTime = Numeric,          # 30 (0-)
        #CloudDasAddress = Literal[256],  #"$DASURL", $ENV
        #CloudServiceGroupId = String[32]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

DOMAIN 절 이름

- 종류: String
- 범위: 31자 이내
- WebtoB 도메인 이름을 설정한다.

DomainID

- 종류: Numeric
- 범위: 0 ~ 255
- 도메인 ID를 설정한다.
- 설정한 ID는 WebtoB 내부 통신에서 도메인을 구분할 때 사용한다.
- 254와 255는 다음과 같은 특별한 의미로 사용된다.

설정값	설명
254	"304 Not Modified"를 항상 Worker Thread에서 처리한다. 따라서 304 응답에 대한 처리가 느릴 수 있으므로 권장하지 않는다.
255	HTH에서 conditional-get 요청에 대해 항상 "304 Not Modified"로 응답한다. 따라서 리소스가 수정되었더라도 브라우저나 Proxy에서는 오래된 리소스를 그대로 사용하게 되므로 권장하지 않는다.

MaxSvc

- 종류: Numeric
- 범위: 1 ~ 65535(0xffff)
- 기본값: 512
- 설정 가능한 최대 서비스의 수를 변경한다.
- 서비스는 SERVICE 절 뿐만 아니라, URI 절 및 EXT 절도 포함한다(URI 절 및 EXT 절 설정은 자동으로 SERVICE 절에 등록된다).

NHthChkTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 30
- HTH가 살아있는지 확인하기 위해 체크하는 시간을 설정하며 HTH가 블록이 걸려있는지 확인하는데 사용된다.
- NHthChkTime을 설정하고 연속된 2회의 요청에도 응답이 없으면 해당 HTH에 이상이 발생했다고 인식하고 해당 HTH를 재시작한다.

CloudDasAddress

- 종류: Literal
- 범위: 255자 이내
- Cloud 환경에서 WebtoB와 JEUS를 연동하는 경우 Auto Scale을 위해 서로 이벤트를 감지할 수 있도록 DAS의 IP 주소와 포트 번호를 설정한다.
- 환경변수를 사용할 수 있으며 이 경우 Cloud용 이미지를 만들 때 편리하다.

CloudServiceGroupId

- 종류: String
- 범위: 31자 이내
- Cloud 환경에서 업무별로 WebtoB를 구분할 수 있는 ID를 설정한다.

3.2.2. 예제

다음은 DOMAIN 절을 설정한 예제이다.

```
*DOMAIN
webtob
```

3.3. NODE 절

NODE 절은 WebtoB 노드에 대한 구체적인 환경에 대해 설정한다. NODE 절은 노드의 전체적인 동작과 관련된 설정을 정의한다. VHOST 절을 설정하여 가상 호스트 설정한 경우 NODE 절 설정은 기본 호스트로 동작한다.

NODE 절에는 다음과 같은 내용들이 정의될 수 있다.

- WebtoB 시스템 경로
- 서비스할 문서들이 위치한 최상위 경로 (document root)
- 공유 메모리의 Key 값
- 서비스 IP 및 포트 번호 설정
- 기타 시스템 전체적인 설정

3.3.1. 설정 항목

다음은 NODE 절의 환경설정 형식이다.

```
*NODE
name   # String[128]
       WebtobDir = Literal[256],            # $ENV
       ShmKey = Numeric,                    # (32768-262143)/(0x8000-0x3FFFF)
       #Nodename = Literal[128],            # "<*NODE.Name>", $ENV
       #User = String[32],
```

```
        #Group = String[32],
        #Hostname = Literal[128],           # "<OS hostname>"
        #DocRoot = Literal[256],            # "docs/", $ENV, R.PATH
        #SvrRoot = Literal[256],            # "$WEBTOBDIR", $ENV
        #Method = Literal[256],             # "GET,POST,HEAD,OPTIONS", PM.LIST
        #Hth = Numeric,                     # 1 (1-100)
        #HthQTimeout = Numeric,             # 0 (0-)f
        #Port = Literal[1024],              # "80" or "443" (1-65535), LIST[100]
        #JsvPort = Numeric,                 # 9999 (1-65535)
        #JsvSslPort = Numeric,              # 0 (1-65535)
        #JsvSslFlag = Boolean,              # N
        #JsvSslName = String[32],
        #RacPort = Numeric,                 # 3333 (1-65535)
        #SslFlag = Boolean,                 # N
        #SslName = String[32],
        #Timeout = Numeric,                 # 300 (1-)
        #InitialConnectionTimeout = Numeric, # 10 (0-)
        #RequestHeaderTimeout = Numeric,    # 60 (0-)
        #RequestBodyTimeout = Numeric,      # 0 (0-)
        #CacheKey = String[32],             # HOST_URI (HOST_URI, REAL_PATH)
        #CacheEntry = Numeric,              # 1024 (0-)
        #MaxCacheMemorySize = Numeric,      # 100 (0-)
        #CacheMaxFileSize = Numeric,        # 8192 (0-)
        #CacheMaxCompressSize = Numeric,    # 0 (0-)
        #CacheRefreshImage = Numeric,       # 3600 (0-)
        #CacheRefreshHtml = Numeric,        # 3600 (0-)
        #CacheRefreshJsv = Numeric,         # 3600 (0-)
        #CacheRefreshRproxy = Numeric,      # 3600 (0-)
        #DiskCache = String[32],
        #Keepalive = Boolean,               # Y
        #KeepaliveTimeout = Numeric,        # 60 (1-)
        #KeepaliveMax = Numeric,            # 0 (0-)
        #MaxUser = Numeric,                 # <auto> (1-)
        #AppDir = Literal[256],             # $ENV, R.PATH
        #PathDir = Literal[256],            # "path/", $ENV, R.PATH
        #UsrLogDir = Literal[256],          # "log/usrlog/", $ENV, R.PATH
        #IconDir = Literal[256],            # $ENV, R.PATH
        #UserDir = Literal[256],            # MULTI
        #IndexName = Literal[256],          # "index.html", LIST
        #DirIndex = Literal[32],
        #Options = Literal[256],            # "HTML,CGI,SSI,PHP,JSV", PM.LIST
        #ErrorDocument = Literal[256],      # LIST[64]
        #Logging = Literal[256],            # LIST[4]
        #ErrorLog = Literal[256],           # LIST
        #Filter = Literal[256],             # LIST[64]
        #Listen = Literal[1024],            # LIST[100]
        #IpcPerm = Numeric,                 # 0700 (0600-0777)
        #ListenBacklog = Numeric,           # <auto> (1-)
        #SendBufferSize = Numeric,          # <auto> (0-)
        #RecvBufferSize = Numeric,          # <auto> (0-)
        #IpcSendBufferSize = Numeric,       # <auto> (0-)
        #IpcRecvBufferSize = Numeric,       # <auto> (0-)
        #HthIpcSendBufferSize = Numeric,    # <auto> (0-)
        #HthIpcRecvBufferSize = Numeric,    # <auto> (0-)
        #SvrIpcSendBufferSize = Numeric,    # <auto> (0-)
        #SvrIpcRecvBufferSize = Numeric,    # <auto> (0-)
        #MimetypesConfig = Literal[256],    # "config/mime.types", $ENV, R.PATH
        #DefaultMimetype = Literal[128],    # "text/html"
        #RPAFHeader = Literal[256],
```

```
        #TimeOutStatus = Numeric,            # 500 (511-599)
        #Expires = Literal[256],              # LIST[64]
        #LimitRequestBody = Numeric,          # 0 (0-)
        #LimitRequestFields = Numeric,        # 100 (0-)
        #LimitRequestFieldSize = Numeric,     # 8190 (0-16382)
        #LimitRequestLine = Numeric,          # 8190 (0-16382)
        #ServerTokens = Literal[256],         # "Off"
        #ServiceOrder = Literal[256],         # "uri,ext"
        #IPCBasePort = Numeric,               # 6666 (1-65535)
        #TcpGW = Literal[1024],               # LIST[64]
        #DefaultCharset = Literal[32],        # "Off"
        #JsvAccessName = String[32],
        #UseEtag = Boolean,                   # Y
        #TerminateCgiUponClientClose = Boolean, # Y
        #URLRewrite = Boolean,                # N
        #URLRewriteConfig = Literal[256],     # R.PATH
        #LogPerm = Numeric,                   # 0600 (0600-0777)
        #SysLog = Literal[256],               # "default_syslog"
        #KeepAliveErrorStatusCode = Literal[256], # (HTTP status code), LIST
        #MaxDechunkSize = Numeric,            # 10485760(10MB) (0-)
        #HtmlsForbidsWEBINF = Boolean,        # Y
        #HtlHthSendSocketBufferSize = Numeric, # 0 (0-)
        #HtlHthRecvSocketBufferSize = Numeric, # 0 (0-)
        #TraceAccessLog = Boolean,            # N
        #CheckURL = Boolean,                  # N
        #CheckURLFrom = Literal[32],          # "utf-8"
        #CheckURLTo = Literal[32]
        #CheckUrlJsvExcept = Boolean,         # N
        #SSIMaxDepth = Numeric,               # 16 (1-)
        #ForceCacheModificationCheck = Boolean, # N
        #DebugHTHMemory = Boolean,            # N
        #Headers = Literal[256],              # LIST[15]
        #DOSBlock = Boolean,                  # N
        #DOSBlockTableSize = Numeric,         # 3097 (1-)
        #DOSBlockPageCount = Numeric,         # 5 (0-)
        #DOSBlockPageInterval = Numeric,      # 1 (1-)
        #DOSBlockSiteCount = Numeric,         # 50 (0-)
        #DOSBlockSiteInterval = Numeric,      # 1 (1-)
        #DOSBlockPeriod = Numeric,            # 30 (1-)
        #DOSBlockWhiteList = Literal[256],    # LIST[256], MULTI
        #BindIPv6Only = Boolean,              # <OS default>
        #JsvListen = Literal[1024],           # LIST[10]
        #JsvSslListen = Literal[1024],        # LIST[10]
        #UpperDirRestrict = Boolean,          # N
        #CheckPingTimeoutStatus = Numeric,    # 503 (511-599)
        #IgnoreMissingColonErr = Boolean,     # N
        #HTTP2Flag = Boolean                  # N
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

NODE 절 이름

- 종류: String

- 범위: 127자 이내
- NODE 이름을 설정한다.

WebtobDir(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- WebtoB가 설치된 경로를 설정한다.
- 환경변수를 사용할 수 있으며 $WEBTOBDIR로 설정하면 편리하다.

ShmKey(필수 항목)

- 종류: Numeric
- 범위: 32768 ~ 262143 / 0x8000 ~ 0x3FFFF
- 공유 메모리 세그먼트(Shared Memory Segment)를 설정한다. WebtoB 내부 프로세스는 서로 정보를 공유하기 위해 공유 메모리를 사용한다.
- 다음의 3개 key가 사용된다.
  - ShmKey
  - (ShmKey + 1)
  - (ShmKey + 2)

  공유 메모리 Key 값을 정의하기 전에 이 Key 값들이 다른 프로그램 또는 다른 업무에서 사용되는지 반드시 확인해야 한다. 그렇지 않으면 WebtoB가 기동될 때 이 프로그램과 충돌을 일으켜서 실행이 되지 않는다.

UNIX/Linux의 **"ipcs"** 명령을 사용하여 공유 메모리를 확인할 수 있다. WebtoB를 강제로 종료시킨 경우 공유 메모리가 반환되지 않고 남아있을 경우에 한해 **"ipcrm"** 명령을 통해 수동으로 반환할 수 있다.

Nodename

- 종류: Literal
- 범위: 127자 이내, 환경변수($NODENAME) 사용 가능
- 기본값: NODE 절 이름을 사용
- 노드의 물리적인 이름으로 운영체제별로 확인된 이름을 정의한다.
  - UNIX/Linux : "uname -n" 혹은 "hostname" 명령
  - Windows : "hostname.exe" 명령
- 설정한 노드 이름은 반드시 파일에 등록되어 있어야 한다.

- UNIX/Linux : /etc/hosts
- Windows : %SystemRoot%\system32\drivers\etc\hosts

User

- 종류: String
- 범위: 31자 이내
- 설정된 계정 권한으로 WebtoB가 동작하도록 한다.

UNIX/Linux 환경에서만 설정할 수 있으며, 시스템 보안을 위해 root가 아닌 일반 계정으로 설정할 것을 권장한다.

Group

- 종류: String
- 범위: 31자 이내
- 설정된 그룹 권한으로 WebtoB가 동작하도록 한다.

UNIX/Linux 환경에서만 설정할 수 있으며, 시스템 보안을 위해 root가 아닌 일반 그룹으로 설정할 것을 권장한다.

Hostname

- 종류: Literal
- 범위: 127자 이내
- 기본값: 시스템 호스트 이름을 사용
- 호스트 이름을 설정한다.
- CGI/PHP에서 "SERVER_NAME" 변수로 사용하며, "Host" 필드가 없는 HTTP/1.0 요청을 처리할 때 이 설정값을 대신 사용한다.

DocRoot

- 종류: Literal
- 범위: 255자 이내
- 기본값: "docs/"
- 서비스할 문서가 위치한 최상위 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여

설정된다.

- 다음 지시자를 사용해서 디렉터리 설정 패턴을 동적으로 설정할 수 있다.

지시자	설명
%p	요청한 포트 번호로 치환한다.
%n	Hostname이나 IP 주소의 n번째 요소로 치환한다. 만약 n을 0으로 하면 전체 문자열이 사용된다. 마이너스 기호(-)가 앞에 오면 Hostname이나 IP 주소의 끝에서부터 센다. 플러스 기호(+)가 뒤에 오면 Hostname이나 IP 주소의 나머지가 사용된다.
%n.m	n번째 요소의 m번째 문자로 치환한다. 위와 같이 마이너스 기호(-)나 플러스 기호(+)가 붙을 수 있다.
%%	단일 퍼센트(%) 표시로 치환한다.

지시자를 사용할 경우 DocRoot를 절대 경로로 설정해야 한다. 환경변수를 사용했을 경우 치환된 경로가 최종적으로 절대 경로일 경우에만 위 지시자를 사용할 수 있다.

SvrRoot

- 종류: Literal
- 범위: 255자 이내
- 기본값: "$WEBTOBDIR"
- ALIAS 절의 RealPath를 상대 경로로 설정했을 경우 루트 디렉터리로 사용할 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

Method

- 종류: Literal
- 범위: 255자 이내
- 기본값: "GET,POST,HEAD,OPTIONS"
- 처리할 수 있는 HTTP Request 메소드를 설정한다. 클라이언트로부터 받은 Request 메소드가 설정되어 있지 않으면 해당 요청은 처리되지 않는다.
- GET, POST, HEAD, OPTIONS, CONNECT, TRACE, PROPFIND, PUT, DELETE, MKCOL, COPY, MOVE를 지원한다.
- 특정 메소드를 서비스하지 않을 경우 "–OPTIONS"와 같이 하이픈(-)을 메소드 이름 앞에 설정한다.

Hth

- 종류: Numeric
- 범위: 1 ~ 100

- 기본값: 1
- HTH 프로세스의 수를 설정한다.
- HTH 프로세스 하나가 처리할 수 있는 동시 접속자의 수가 제한되어 있는데, 이를 늘리는 경우 원하는 적당한 값을 설정한다.
- HTH 프로세스 하나당 처리할 수 있는 사용자 수는 **"wscfl"** 명령의 출력 결과나 **"wsboot"**를 실행하는 경우 출력 결과를 통해 확인할 수 있다.

```
$ wscfl -i http.m
Current configuration:
        Number of client handler(HTH) = 4
        Supported maximum user per node = 32552
        Supported maximum user per handler = 8138
```

HthQTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 서버 큐에 대기하고 있는 사용자 요청의 타임아웃을 설정한다.
- 사용자 요청이 많아 해당 요청을 처리할 서버 프로세스가 없을 경우 해당 요청은 서버 큐에서 요청을 처리할 서버 프로세스가 생길 때까지 기다리게 된다. 이때, 설정된 시간보다 오래 기다리고 있는 요청은 큐에서 빼고 "503 Service Unavailable"로 응답한다.

Port

- 종류: Literal
- 범위: 1023자 이내, 1 ~ 65535
- 기본값: "80" 또는 "443" (SSLFlag를 'Y'로 설정하는 경우에는 "443"을 기본값으로 사용한다.)
- 사용자가 접속할 수 있는 서비스 포트를 설정한다.
- 콤마(,)로 구분하여 100개까지 설정 가능하다.

Port와 Listen 항목이 같이 설정된 경우 Port 설정은 무시된다.

JsvPort

- 종류: Numeric
- 범위: 1 ~ 65535

- 기본값: 9999
- WebtoB와 JEUS를 연동하여 사용할 때 필요한 서비스 포트를 설정한다.

  WebtoB가 해당 포트를 열면 JEUS가 "WEBMain.xml"에 다음과 같은 설정을 통하여 연결한다.

```
<?xml version="1.0"?>
<web-container xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="6.0">
    <context-group>
        <group-name>MyGroup</group-name>
        <webserver-connection>
            <webtob-listener>
                <listener-id>webtob1</listener-id>
                <port>9900</port>
                <output-buffer-size>8192</output-buffer-size>
                <thread-pool>
                   <min>5</min>
                   <max>5</max>
                   <step>5</step>
                   <max-idle-time>30000</max-idle-time>
                </thread-pool>
                <webtob-address>__webtob_address__</webtob-address>
                <registration-id>__webtob_server_name__</registration-id>
                <disable-pipe>true</disable-pipe>
            </webtob-listener>
        </webserver-connection>
    </context-group>
</web-container>
```

JsvSslPort

- 종류: Numeric
- 범위: 1 ~ 65535
- WebtoB와 JEUS를 SSL 통신으로 연동하여 사용할 때 필요한 서비스 포트를 설정한다.

  WebtoB가 해당 포트를 열면 JEUS가 "WEBMain.xml"에 다음과 같은 설정을 통하여 연결한다.

```
<?xml version="1.0"?>
<web-container xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="6.0">
    <context-group>
        <group-name>MyGroup</group-name>
        <webserver-connection>
            <webtob-listener>
                <listener-id>webtob1</listener-id>
                <port>10443</port> <!-- JsvSslPort -->
                <output-buffer-size>8192</output-buffer-size>
                <thread-pool>
                   <min>5</min>
                   <max>5</max>
                   <step>5</step>
                   <max-idle-time>30000</max-idle-time>
                </thread-pool>
                <webtob-address>__webtob_address__</webtob-address>
```

```
                <registration-id>__webtob_server_name__</registration-id>
                <disable-pipe>true</disable-pipe>
                   <secure>
                      <!-- WebtoB *SSL절에 설정한 서버인증서를
                           Java KeyStore(JKS) 형식으로 만들어서 지정 -->
                      <trust-store-file-path>
                          __server_certstore.jks__
                      </trust-store-file-path>
                      <trust-store-file-password>
                          __xxxx__
                      </trust-store-file-password>
                   </secure>
            </webtob-listener>
         </webserver-connection>
    </context-group>
</web-container>
```

JsvSslFlag

- 종류: Boolean
- 기본값: N
- JEUS 연결을 위한 포트를 SSL/TLS 프로토콜을 사용하여 서비스할지 여부를 설정한다.
- JsvSslName 설정으로 적용할 SSL 절 항목을 지정할 수 있다.

JsvSslName

- 종류: String
- 범위: 31자 이내
- JsvSslFlag를 'Y'로 설정하는 경우 적용되며, 사용할 SSL 절 항목을 설정한다.

RacPort

- 종류: Numeric
- 범위: 1 ~ 65535
- 기본값: 3333
- WebtoB 관리 프로세스 중 하나인 **wsracd** 데몬이 서비스를 위해 사용하는 포트를 설정한다.

SSLFlag

- 종류: Boolean
- 기본값: N

- 서비스 포트를 SSL/TLS 프로토콜을 사용하여 서비스할지 여부를 설정한다.
- SSLName 설정으로 적용할 SSL 절 항목을 지정할 수 있다.

SSLName

- 종류: String
- 범위: 31자 이내
- SSLFlag를 'Y'로 설정하는 경우 적용되며, 사용할 SSL 절 항목을 설정한다.

Timeout

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX
- 기본값: 300
- 사용자가 연결한 소켓에서 데이터를 읽거나 쓸 때 적용하는 타임아웃을 설정한다.
- 사용자 요청을 처리하고 있을 때 적용되는 설정이며, 해당 시간 동안 사용자가 소켓에 데이터를 쓰지 않거나 소켓으로부터 데이터를 읽지 않는 경우 해당 소켓을 닫는다.

InitialConnectionTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 10
- 사용자로부터의 첫 번째 요청을 기다리기 위한 타임아웃을 설정한다.
- 사용자가 TCP 연결을 맺은 후 어떠한 요청도 보내지 않고 있을 때 적용되는 설정이며, 해당 시간 동안 사용자가 어떠한 요청도 보내지 않는다면 해당 소켓을 닫는다. 사용자가 요청을 보내고 있는 중이거나 이미 요청을 한 번 이상 보낸 경우는 해당 되지 않는다.
- SSL를 연결할 때에도 적용되며, 해당 시간 동안 사용자가 SSL 연결을 완료하지 못하거나, SSL 연결을 완료하였더라도 어떠한 요청을 보내지 않는다면 해당 소켓을 닫는다.
- 0으로 설정한 경우 사용자로부터의 첫 번째 요청을 기다리기 위한 시간을 제한하지 않는다.

RequestHeaderTimeout

- 종류: Numeric

- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 60
- HTTP Request Header를 기다리기 위한 타임아웃을 설정한다.
- 사용자가 요청을 보내고 있을 때 적용되는 설정이며, 해당 시간 동안 사용자가 HTTP Request Header를 모두 보내지 않은 경우 "408 Request Timeout"으로 사용자에게 응답한다.
- Pipelined Request인 경우(또는 앞선 요청에 대한 응답이 끝나지 않은 상태에서 다음 요청이 들어온 경우) 다음 요청에 대한 시작시간은 앞선 요청에 대한 응답이 끝난 직후로 한다.
- 0으로 설정한 경우 사용자가 HTTP Request Header를 보내는 시간을 제한하지 않는다.

RequestBodyTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0
- HTTP Request Body가 있는 경우 Body를 기다리기 위한 타임아웃을 설정한다.
- 사용자가 요청을 보내고 있을 때 적용되는 설정이며, 해당 시간 동안 사용자가 HTTP Request Body를 다 보내지 않은 경우 "408 Request Timeout"으로 사용자에게 응답한다.
- 0으로 설정한 경우 사용자가 HTTP Request Body를 보내는 시간을 제한하지 않는다.

CacheKey

- 종류: String
- 범위: 31자 이내
- 기본값: HOST_URI
- 캐싱하기 위한 Key 값을 생성할 때 사용할 값을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
HOST_URI	Request Header의 HOST 값과 Request URI를 사용하여 Key를 생성한다. 동일한 REQUEST_URI라 하더라도 HTTP_HOST에 따라 VHOST가 달라진다. 이는 실제 경로가 달라질 수 있음을 의미한다. 하지만 VHOST가 다르다 하더라도 실제 경로가 같다면 중복으로 캐싱되는 문제가 발생할 수 있다. 이는 동일한 요청에 대해서 HOST에 따라 다르게 캐싱되어져 있는 파일로 응답할 수 있음을 의미한다. VHOST별 DocRoot가 다르다면 이 값(HOST_URI)를 사용하는 것이 낫다. 실제 경로(Real Path)를 계산하지 않으므로 REAL_PATH 방식보다 응답 속도가 빠르다. JEUS에서 처리되거나 Reverse Proxy로 처리되는 응답은 실제 경로를 계산할 수 없으므로 모두 이 값을 사용한다.
REAL_PATH	요청 URI의 실제 경로를 사용하여 Key를 생성한다. 이 값은 SVRTYPE이 HTML인 경우에만 적용되고, HOST_URI를 사용할 경우 발생할 수 있는 문제점을 보완한다. HTTP_HOST가 다르더라도 동일한 요청한 파일의 실제 경로가 같을 수 있다. 이 같은 경우 HTTP_HOST에 상관없이 하나의 파일만 캐싱되므로 메모리가 불필요하게 사용되지 않는다. 실제 경로를 계산해야 하므로 HOST_URI를 Key로 사용하는 것보다 응답 속도가 조금 느릴 수 있다.

CacheEntry

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 128
- HTH 캐시(Cache)의 Hash 테이블 키의 크기를 설정한다.
- 0으로 설정하는 경우 응답을 캐시하지 않는다.

MaxCacheMemorySize

- 종류: Numeric
- 단위: Mbytes
- 범위: 0 ~ INT_MAX
- 기본값: 100
- HTH 프로세스가 캐시를 위해 사용하는 최대 메모리 사이즈를 설정한다. HTH별로 각각 캐시하기 때문에 해당 설정은 HTH 프로세스별로 각각 적용되는 값이다(모든 HTH 캐시가 사용하는 총 메모리 사이즈가 아니다).
- 최대 메모리 사이즈를 넘는 경우 클라이언트가 잘 접근하지 않는 캐시 항목을 일정 부분 삭제하며 syslog를 통해 관리자에게 알려준다.
- 0으로 설정하는 경우 응답을 캐시하지 않는다.

CacheMaxFileSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 8192
- 캐시할 수 있는 응답(Response Header + Response Body) 하나의 최대 사이즈를 설정한다.
- 0으로 설정하는 경우 응답을 캐시하지 않는다.

CacheMaxCompressSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 캐시할 수 있는 압축된 응답(Response Header + Response Body) 하나의 최대 사이즈를 설정한다.
- 0으로 설정하는 경우 압축된 응답을 캐시하지 않는다.

CacheRefreshHtml

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 3600
- 캐시된 응답 중 "Content-Type"이 "text/html"인 응답에 대한 유효시간을 설정한다(SVRTYPE이 HTML인 경우). 즉, "text/html" 응답은 캐시된 후 설정된 시간(초) 동안만 유효하다.
- 0으로 설정하는 경우 캐시된 응답은 만료시간 없이 유효하다고 판단한다.
- Conditional-GET 요청(또는 ForceCacheModificationCheck 설정에 의한 요청)은 캐시된 응답이 변경되었는지 체크한다. 응답이 변경된 경우 캐시된 응답은 삭제 후 변경된 응답으로 갱신된다.

CacheRefreshImage

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX

- 기본값: 3600
- 캐시된 응답 중 "Content-Type"이 "text/html"이 아닌 응답에 대한 유효시간을 설정한다(SVRTYPE이 HTML인 경우). 즉, "text/html"이 아닌 응답은 캐시된 후 설정된 시간(초) 동안만 유효하다.
- 0으로 설정하는 경우 캐시된 응답은 만료시간 없이 유효하다고 판단한다.
- Conditional-GET 요청(또는 캐시된 응답은 만료시간 없이 유효하다고 판단한다 설정에 의한 요청)은 캐시된 응답이 변경되었는지 체크한다. 응답이 변경된 경우 캐시된 응답은 삭제 후 변경된 응답으로 갱신된다.

CacheRefreshJsv

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 3600
- JEUS로부터 받은 응답을 캐시하게 되는 경우 캐시되는 응답의 유효시간을 계산하기 위한 값을 설정한다.
- 클라이언트의 요청에 대해 WebtoB가 캐시된 내용에서 응답할 때는 캐시된 내용이 유효한지를 판단한다. 캐시된 내용이 유효한지 여부는 설정된 값(유효시간)을 기준으로 판단한다.
  - 첫 번째로 캐시되어질 당시의 Response Header에 Cache-Control:max-age 값이 있으면 max-age(초) 동안 유효하다고 설정한다.
  - 두 번째로 캐시되어질 당시의 Response Header에 Expires 값이 있으면 Expires(시간)까지 유효하다고 설정한다.
  - 세 번째로 Response Header에 위 두 가지 Header가 없이 캐시되는 경우 CacheRefreshJsv(초) 동안 유효하다고 설정한다. 즉, Response Header에 Cache-Control:max-age, Expires 값이 없는 경우 JEUS로부터 받은 응답이 캐시된 후 캐시된 내용은 CacheRefreshJsv(초) 동안 유효하다.
- 유효시간을 설정할 때 우선순위는 다음의 순서로 적용된다.

```
Cache-Control:max-age > Expires > CacheRefreshJsv
```

- 0으로 설정하는 경우(Response Header 값의 Cache-Control:max-age, Expires 값이 없다면) 캐시된 응답은 만료시간 없이 유효하다고 판단한다. 즉, 0으로 설정하는 경우(Response Header 값의 Cache-Control:max-age, Expires 값이 없다면) Conditional-GET 요청에 의해 다시 JEUS로부터 변경된 응답을 받은 경우에만 refresh된다.

CacheRefreshRproxy

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 3600

- Reverse Proxy로 처리된 응답을 캐시하게 되는 경우 캐시되는 응답의 유효시간을 계산하기 위한 값을 설정한다.
- 클라이언트의 요청에 대해 WebtoB가 캐시된 내용에서 응답할 때는 캐시된 내용이 유효한지를 판단한다. 캐시된 내용이 유효한지 여부는 설정된 값(유효시간)을 기준으로 판단한다.
  - 첫 번째로 캐시되어질 당시의 Response Header에 Cache-Control:max-age 값이 있으면 max-age(초) 동안 유효하다고 설정한다.
  - 두 번째로 캐시되어질 당시의 Response Header에 Expires 값이 있으면 Expires(시간)까지 유효하다고 설정한다.
  - 세 번째로 Response Header에 위 두 가지 Header가 없이 캐시되는 경우 CacheRefreshRproxy(초) 동안 유효하다고 설정한다. 즉, Response Header에 Cache-Control:max-age, Expires 값이 없는 경우 Reverse Proxy로 처리된 응답이 캐시된 후 캐시된 내용은 CacheRefreshRproxy(초) 동안 유효하다.
- 유효시간을 설정할 때 우선순위는 다음의 순서로 적용된다.

```
Cache-Control:max-age > Expires > CacheRefreshRproxy
```

- 0으로 설정하는 경우(Response Header 값의 Cache-Control:max-age, Expires 값이 없다면) 캐시된 응답은 만료시간 없이 유효하다고 판단한다. 즉, 0으로 설정하는 경우(Response Header 값의 Cache-Control:max-age, Expires 값이 없다면) Conditional-GET 요청에 의해 다시 Reverse Proxy로 처리되어 변경된 응답을 받은 경우에만 refresh된다.

DiskCache

- 종류: String
- 범위: 31자 이내
- 사용할 DISK_CACHE 절 항목을 설정한다.

Keepalive

- 종류: Boolean
- 기본값: Y
- Keepalive(HTTP persistent connection) 사용 여부를 결정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	사용자는 연결을 재사용하여 한 번의 연결로 여러 개의 요청을 처리할 수 있다.
N	사용자는 요청을 처리하기 위해 매번 소켓을 다시 연결해야 한다.

KeepaliveTimeout

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX
- 기본값: 60
- Keepalive를 유지하는 시간을 설정한다.
- 사용자 요청 처리가 끝난 후 설정된 시간이 지나면 연결을 끊는다.

KeepaliveMax

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- Keepalive 요청 건수를 제한할 경우 설정한다.
- 0으로 설정하는 경우 요청 건수를 제한하지 않는다.

MaxUser

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 라이선스 및 환경설정에 따라 다르며, 보통 7000 ~ 8000 정도이다.
- 동시 접속 가능한 최대 사용자 수를 설정한다.
- MaxUser 계산하는 방법은 다음과 같다.

```
MaxUser= MIN(<운영체제의 프로세스당 openfile 제한>, <WebtoB의 FD 제한, 보통 16384>) *
         <HTH 수> - <내부 통신에 사용하는 FD 수, SERVER 절의 MaxProc의 총합 +
         HTH 수 + 2(WSM, HTL)>
```

AppDir

- 종류: Literal
- 범위: 255자 이내
- WBAPI를 사용하여 작성된 서버의 실행 파일이 위치한 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

SVRGROUP 절에 AppDir 설정이 있을 경우 SVRGROUP의 설정을 사용한다. 즉, NODE 절의 설정은 기본 AppDir을 의미한다.

PathDir

- 종류: Literal
- 범위: 255자 이내
- 기본값: "path/"
- WebtoB를 운영하는 경우 내부 통신에 사용하는 named-pipe 및 각 프로세스들의 pid를 기록하는 "webtob.pid" 파일을 생성할 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

UsrLogDir

- 종류: Literal
- 범위: 255자 이내
- 기본값: "log/usrlog/"
- 사용자 로그를 기록할 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

UsrLogDir은 SERVER, SVRGROUP, NODE 절 순으로 먼저 설정된 값을 사용한다.

IconDir

- 종류: Literal
- 범위: 255자 이내
- 디렉터리 인덱싱 기능에서 사용하는 아이콘이 위치한 경로를 설정한다.
- 해당 설정값은 html 태그에서 사용하는 경로가 되기 때문에 슬래시(/)와 DocRoot를 기준으로 하는 상대 경로로 설정해야 한다.

UserDir

- 종류: Literal
- 범위: 255자 이내
- 사용자별 디렉터리를 설정할 경우 사용한다.
- 다음의 3가지로 설정할 수 있다.
  - *<per-user-home-directory>*

    - disabled [<*user-list*>]
    - enable [<*user-list*>]
- 다음은 "~/public_html"을 사용자별 디렉터리로 설정하고, user1, user2, user3만 사용자별 디렉터리를 허용하도록 설정하는 경우의 예이다.

```
*NODE
mynode
    ...,
    UserDir = "public_html",
    UserDir = "disabled",    # globally disabled
    UserDir = "enable user1 user2 user3",
    ...
```

IndexName

- 종류: Literal
- 범위: 255자 이내
- 기본값: "index.html"
- 디렉터리에 대한 요청을 하는 경우 기본으로 처리할 문서의 이름을 설정한다. 예를 들어 클라이언트가 보낸 요청이 "/somedir/"이고, 해당 경로가 디렉터리인 경우 "/somedir/index.html"을 서비스한다.

DirIndex

- 종류: Literal
- 범위: 31자 이내
- 적용할 DIRINDEX 절의 이름을 설정한다.

디렉터리 인덱싱 기능을 사용하기 위해서는 Options 설정에 "INDEX"를 추가해야 하며, DIRINDEX 절 설정을 통해 디렉터리 인덱싱 기능의 세부 동작 방식을 변경할 수 있다.

Options

- 종류: Literal
- 범위: 255자 이내
- 기본값: "HTML,CGI,SSI,PHP,JSV,USER"
- WebtoB가 제공하는 서비스와 기타 작동 방식에 대한 설정을 한다.
- 특정 옵션을 사용하지 않을 경우 옵션 이름 앞에 하이픈(-)을 설정한다. 예를 들어 "-CGI"는 WebtoB가 CGI 서버를 사용하지 않겠다는 의미이다. 이때, CGI 서버가 필요한 클라이언트 요청은 거절된다.
- 다음은 설정 가능한 옵션에 대한 설명이다.

옵션	설명
HTML	HTML 서비스를 처리한다.
CGI	CGI 서버를 사용한다.
PHP	PHP 서버를 사용한다.
SSI	SSI 서버를 사용한다.
JSV	JSV(JEUS) 서버를 사용한다.
INDEX	클라이언트가 특정 파일 이름을 지정하지 않고 디렉터리에 요구를 보낼 때 디렉터리의 내용을 보여준다. 예를 들어 "/docroot/dir/" 요청이 오면 해당 디렉터리에 존재하는 파일 이름과 정보가 클라이언트로 전달된다.
ALL	"HTML, CGI, PHP, SSI, JSV, INDEX"와 같다.
StrictAuthorizationHdrFormat	HTTP 요청에 대하여 AUTHENT 절 설정을 통해 인증처리를 하는 경우 Authorization header field의 포맷을 검사한다. HTTP 요청의 Authorization header field의 포맷이 HTTP 1.1에 정의된 포맷에 맞지 않거나, 인증타입이 "Basic"이나 "Digest"가 아닌 경우 요청을 보낸 사용자에게 에러로 응답한다. 하이픈(-) 설정인 경우 Authorization header field의 포맷을 검사하여 인증타입이 없는 경우 인증타입을 "Basic"으로 간주하여 처리한다. 예를 들어 Authorization header field의 값이 인증타입 없이 "YXV0aDIucGFzczp0bWF4MTIzNA=="인 경우 하이픈(-) 설정을 했다면 인증타입을 "Basic"으로 간주하여 인증처리를 하고, 하이픈(-) 설정을 하지 않았다면 인증처리 없이 사용자에게 에러로 응답한다.
IgnoreExpect100Continue	HTTP/1.1을 요청할 때 요청 헤더에 "Expect: 100-continue"가 포함된 경우 이를 무시하도록 한다. 이 경우에는 "100 Continue"로 응답하지 않는다.
ProcessIncompleteRequest	클라이언트가 보낸 POST 요청의 body 데이터가 Content-Length만큼 다 도착하지 않았더라도, 클라이언트로부터 받은 body 데이터를 해당 서버로 다 보내도록 한다.
ExcludeAllowHeaderOnError	*NODE.Method 설정을 통해 disable 된 Method 요청에 대해서 error 응답이 나갈 경우, 응답에 "Allow" Header를 포함하지 않는다.

ErrorDocument

- 종류: Literal
- 범위: 255자 이내
- HTTP 에러 페이지를 사용자가 지정한 페이지로 대신 사용할 경우 ERRORDOCUMENT 절에 정의한 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

Logging

- 종류: Literal
- 범위: 255자 이내
- 액세스 로그에 해당되는 LOGGING 절 이름을 설정한다.
- 콤마(,)로 구분하여 4개까지 설정할 수 있다.

ErrorLog

- 종류: Literal
- 범위: 255자 이내
- 에러 로그에 해당되는 LOGGING 절 이름을 설정한다.

Filter

- 종류: Literal
- 범위: 255자 이내
- 사용할 FILTER 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.
- NODE 절 Filter를 설정하면 모든 요청에 대해 NODE 절 Filter가 적용된다.

Listen

- 종류: Literal
- 범위: 1023자 이내
- 특정 IP에만 서비스 포트를 열고 싶을 경우 "<IP>:<Port>"로 설정한다.
- 콤마(,)로 구분하여 100개까지 설정할 수 있다.
- Port 설정보다 우선되며, Listen을 설정한 경우 Port 설정은 무시된다.

IPCPerm

- 종류: Numeric
- 범위: 0600 ~ 0700
- 기본값: 0700
- WebtoB 내부 프로세스(WSM, HTL, HTH, HTMLS, CGIS 등)의 접근권한을 설정한다.

UNIX/Linux 환경에서만 사용할 수 있다.

ListenBacklog

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 511 (UNIX/Linux), <OS default> (Windows)
- 서비스 포트 연결 큐(listen backlog)의 크기를 설정한다. 즉, 서비스 포트에서 동시에 처리할 수 있는 연결 시도 중인 소켓의 수를 제한하는 설정이다.

netstat 명령에서 SYN_RECVD 상태인 소켓의 수와 관련이 있을 수 있다.

SendBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- TCP 전송 버퍼의 크기를 설정한다.

RecvBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- TCP 수신 버퍼의 크기를 설정한다.

IpcSendBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- 내부 프로세스들 간의 메시지 전송 버퍼의 크기를 설정한다.

IpcRecvBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- 내부 프로세스들 간의 메시지 수신 버퍼의 크기를 설정한다.

HthIpcSendBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- 내부 프로세스 중 HTH의 메시지 전송 버퍼의 크기를 따로 설정한다.

HthIpcRecvBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- 내부 프로세스 중 HTH의 메시지 수신 버퍼의 크기를 따로 설정한다.

SvrIpcSendBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: OS default
- 내부 프로세스 중 서버 프로세스의 메시지 전송 버퍼의 크기를 따로 설정한다.

SvrIpcRecvBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX

- 기본값: OS default
- 내부 프로세스 중 서버 프로세스의 메시지 수신 버퍼의 크기를 따로 설정한다.

MimetypesConfig

- 종류: Literal
- 범위: 255자 이내
- 기본값: "config/mime.types"
- MIME-Type과 확장자를 매핑하는 MIME-Type 설정 파일의 경로를 설정한다. 절대 경로와 $WEBTOBDIR의 상대 경로로 설정할 수 있다.
- 아무 값도 설정하지 않은 경우(""로 설정)는 해당 기능을 사용하지 않는다.
- MIME-Type 설정 파일은 다음과 같은 형식으로 설정한다.

```
# '#'로 시작하는 줄은 주석으로 인식한다.
# MIME-Type과 여기에 매핑하고자 하는 확장자를 공백으로 구분하여 설정한다.
# 여러 확장자를 공백으로 구분하여 동시에 설정할 수 있다.
# MIME-Type    Extensions
text/html      html htm
image/jpeg     jpeg jpg jpe
```

DefaultMimetype

- 종류: Literal
- 범위: 127자 이내
- 기본값: "text/html"
- MIME-Type을 결정할 수 없는 문서의 Default Content-Type을 설정한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultMimetype을 사용한다. 둘 다 설정되지 않았을 경우 VHOST 절 및 NODE 절 설정을 사용한다.

RPAFHeader

- 종류: Literal
- 범위: 255자 이내
- proxy 등을 거치면서 변경된 Remote IP의 값을 원래 요청이 발생된 host 값으로 셋팅한다.
- RPAFHeader 추가를 통해 사용자가 header 지정이 가능하다.

```
(예: RPAFHeader = "X-Forwarded-For")
```

- RPAFHeader로 설정된 header가 들어올 경우 header의 IP 값으로 Remote IP를 변경한다.

TimeoutStatus

- 종류: Numeric
- 범위: 511~599
- 기본값 : 500
- Time Out이 발생할 경우 해당 설정의 값으로 HTTP Status Code를 설정하여 회신한다.

Expires

- 종류: Literal
- 범위: 255자 이내
- 적용할 EXPIRES 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

LimitRequestBody

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ LONG_MAX
- 기본값: 0
- 사용자 요청에서 HTTP Request Body 크기를 제한할 경우 설정한다.
- HTTP Request Header 중 "Content-Length" 값을 기준으로 판단하며, 설정된 값보다 클 경우 "413 Request Entity Too Large"로 응답을 처리한다.
- HTTP Request Body가 2G(Long type) 이상인 경우는 JEUS 7 이후 버전부터 처리 가능하다.

Chunked-request의 경우 MaxDechunkSize 설정이 적용된다.

LimitRequestFields

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 100

- 사용자 요청에서 HTTP Request Header field 수를 제한할 경우 설정한다.
- 0으로 설정하는 경우 Header field의 수를 체크하지 않는다.

LimitRequestFieldSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ 16382
- 기본값: 8190
- 사용자 요청에서 HTTP Request Header field 각각의 크기를 제한할 경우 설정한다.
- 10000bytes를 넘어가는 Request Header를 사용하기 위해서는 "WJPv2"를 이용해야 한다.

LimitRequestLine

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ 16382
- 기본값: 8190
- 사용자 요청에서 HTTP Request line 크기를 제한할 경우 설정한다.
- 10000bytes를 넘어가는 Request line을 사용하기 위해서는 "WJPv2"를 이용해야 한다.

ServerTokens

- 종류: Literal
- 범위: 255자 이내
- 기본값: "Off"
- 사용자에게 보내는 Response Header 중 "Server" 필드의 값을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Off	Response Header에 "Server" 필드를 사용하지 않는다.
Prod[uctOnly]	"WebtoB"
Min[imal]	"WebtoB/4.1.3"
OS	(예: "WebtoB/4.1.3 LINUX-K2.6_x86 libc2.3")
Full	(예: "WebtoB/4.1.3 LINUX-K2.6_x86 libc2.3")
Custom	user-specified-name

ServiceOrder

- 종류: Literal
- 범위: 255자 이내
- 기본값: "uri,ext"
- 사용자 요청을 처리할 서버와 서비스를 결정할 때 URI 절과 EXT 절의 우선순위를 결정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
uri, ext	URI 절 설정을 먼저 확인하고 해당되는 설정이 없는 경우 EXT 절의 설정을 확인한다.
ext, uri	EXT 절 설정을 먼저 확인하고 해당되는 설정이 없는 경우 URI 절의 설정을 확인한다.

URI 절과 EXT 절 모두 해당되지 않을 경우 기본 HTML 서비스(Worker Thread)에서 해당 요청을 처리한다.

IPCBasePort

- 종류: Numeric
- 범위: 1 ~ 65535
- 기본값: 6666
- Windows 환경에서만 적용되며, WebtoB 내부 프로세스 간 통신에 사용하는 base port를 설정한다.

WSM은 (IPCBasePort + 1), HTL은 (IPCBasePort + 2), HTH는 (IPCBasePort + 4 + <hth-id>)를 사용한다.

TcpGW

- 종류: Literal
- 범위: 1024자 이내
- 사용할 TCPGW 절 이름을 설정한다.
- 콤마(,)로 구분하거나 여러 번 설정하여 최대 1024개까지 설정할 수 있다.

DefaultCharset

- 종류: Literal

- 범위: 32자 이내
- 기본값: "Off"
- HTTP Response Header 중 "Content-Type" 항목에 추가할 "charset=<value>"의 기본값을 정의한다.

설정값	설명
On	"ISO-8859-1"을 기본값으로 사용한다.
Off	charset을 추가하지 않는다.
<custom-charset>	설정한 <custom-charset>을 기본값으로 사용한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultCharset을 사용한다. 둘 다 설정되지 않았을 경우 VHOST 및 NODE 절 설정을 사용한다.

JsvAccessName

- 종류: String
- 범위: 31자 이내
- JsvPort에 대해 접근 제어를 하는 경우 적용할 ACCESS 절 이름을 설정한다.

UseETag

- 종류: Boolean
- 기본값: Y
- 정적 파일 처리 중 ETag 사용을 결정한다.
- N으로 설정하면 WebtoB는 클라이언트로 전달되는 응답에 ETag를 추가하지 않으며, 클라이언트의 요청에 포함된 ETag는 무시한다.

TerminateCgiUponClientClose

- 종류: Boolean
- 기본값: Y
- CGI, PHP 요청 처리 중 클라이언트가 연결을 끊을 경우 실행 중인 CGI, PHP 프로세스를 강제 중단할지 여부를 결정한다.
- N으로 설정하는 경우 클라이언트가 연결을 끊더라도 CGI, PHP 프로세스가 정상 종료될 때까지 기다린다.

URLRewrite

- 종류: Boolean
- 기본값: N
- URLRewrite 기능을 사용할지 여부를 결정한다.

이 기능을 사용하려면 URLRewriteConfig에 설정 파일의 경로를 지정해야 한다.

URLRewriteConfig

- 종류: Literal
- 범위: 255자 이내
- URLRewirte 기능을 사용하기 위한 설정 파일의 경로를 지정한다. 자세한 설정은 URLRewrite를 참고한다.

LogPerm

- 종류: Numeric
- 범위: 0600 ~ 0777
- 기본값: 0600
- 로그 파일(시스템 로그, 액세스 로그, 에러 로그)의 접근 권한(file access permission)을 설정한다.
- UNIX 계열 운영체제에서 사용하는 파일 접근 권한과 동일한 의미를 가진다. UNIX/Linux 환경에서만 사용할 수 있다.

SysLog

- 종류: Literal
- 범위: 255자 이내
- 기본값: "default_syslog"
- 시스템 로그에 해당하는 LOGGING 절 이름을 설정한다.
- 기본값인 "default_syslog"의 LOGGING 절 FileName 설정은 "log/system_%Y%%M%%D%.log"이다.

KeepAliveErrorStatusCode

- 종류: Literal
- 범위: 255자 이내
- 에러 응답을 보낸 이후 사용자 연결을 유지(persistent connection, keep-alive)하는 경우 해당되는 HTTP Status Code를 설정한다.
- 콤마(,)로 구분하여 여러 Status Code를 설정할 수 있다.

- WebtoB가 사용자 요청에 대해 "304 Not Modified"를 제외한 3xx나 4xx 혹은 5xx HTTP Status Code로 응답할 경우 연결을 유지하지 않고 응답을 보낸 이후 연결을 종료한다.
- 다음은 "302 Found"와 "404 Not Found" 응답을 보낸 후 WebtoB가 클라이언트의 연결을 종료하지 않고 유지하는 예제이다.

```
*NODE
mynode
    ...,
    KeepAliveErrorStatusCode = "302,404",
    ...
```

MaxDechunkSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 10485760
- Chunked 요청을 처리하는 과정에서 de-chunk할 때 허용할 body의 최대 크기를 설정한다.
- 설정된 값보다 chunk body가 큰 요청은 "500 Internal Server Error"로 응답한다.

HtmlsForbidsWEBINF

- 종류: Boolean
- 기본값: Y
- 사용자 요청이 "/WEB-INF/"(case insensitive)를 포함한 요청일 경우 "403 Forbidden"으로 응답하는 기능의 사용 여부를 설정한다.
- 이 기능은 서버와 연결된 애플리케이션 서버(예: JEUS)의 설정 파일들이 클라이언트로 노출되는 경우를 방지하는 역할을 한다.

HtlHthSendSocketBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 WebtoB가 설치된 장비의 OS 기본값을 의미한다.)
- WebtoB 내부 프로세스인 HTL과 HTH가 통신할 때 사용되는 소켓의 버퍼(send buffer) 크기를 설정한다.
- HP-UX 환경에서 sendmsg()의 버그로 인해 추가된 설정이며, UNIX 도메인 소켓(named-pipe)를 통해 sendmsg()를 호출할 때 사용하는 버퍼의 크기를 설정한다.

- HtlHthRecvSocketBufferSize보다 작게 설정한다.

일반적으로 사용자가 설정할 필요는 없다.

HtlHthRecvSocketBufferSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 WebtoB가 설치된 장비의 OS 기본값을 의미한다.)
- WebtoB 내부 프로세스인 HTL과 HTH가 통신할 때 사용되는 소켓의 버퍼(receive buffer) 크기를 설정한다.

일반적으로 사용자가 설정할 필요없다.

TraceAccessLog

- 종류: Boolean
- 기본값: N
- Y로 설정하는 경우 액세스 로그에 요청 처리 과정에 대한 로그를 추가적으로 남긴다.

Y로 설정하는 경우 평소보다 액세스 로그를 2~3배 더 남기게 되므로 주의해야 한다.

- 다음은 저장되는 로그 항목에 대한 설명이다.

로그 항목	설명
[ARRIVED]	요청을 받은 시점
[SEND-TO-SERVER <*type*> <*server-index*>]	요청을 서버에 전달한 시점
[QUEUED <*type*> <*server-name*>]	요청이 서버 큐에 들어간 시점
[JSV SUSPEND FAILOVER]	JSV 서버가 suspend되는 시점에 큐에 있던 요청을 다른 JSV 서버에게 할당한 경우

CheckURL

- 종류: Boolean
- 기본값: N
- HTTP Request URL path의 Charset을 변환할지 여부를 결정한다.

- Charset은 CCS(Coded Character Set) 와 CES(Character Encoding Scheme)의 조합으로, 한글은 일반적으로 EUC-KR이나 UTF-8을 사용한다.
- Request URL path에 사용한 Charset과 서버에서 사용하는 Charset이 다를 경우 CheckURL을 사용해서 Charset을 변환하면 된다.
- Y로 설정한 경우 CheckURLFrom과 CheckURLTo를 설정해야 한다.
- VHOST 절과 함께 설정한 경우 VHOST 절의 설정을 우선한다.

CheckURLFrom

- 종류: Literal
- 범위: 31자 이내
- 기본값: "utf-8"
- HTTP Request URL path가 사용하는 Charset을 설정한다.

"euc-kr"과 "utf-8"만 지원한다.

CheckURLTo

- 종류: Literal
- 범위: 31자 이내
- 서버가 사용하는 Charset을 설정한다.

CheckUrlJsvExcept

- 종류: Boolean
- 기본값: N
- JEUS로 포워딩하는 요청에 대하여 CheckURL 설정을 적용하지 않을지 여부를 결정한다.
- Y로 설정한 경우 JEUS로 요청을 보낼 때에는 URI를 변환하지 않는다.

SSIMaxDepth

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 16
- SSI(Server-side include) 요청을 처리하는 경우 nested-include의 depth 제한을 설정한다.

  SSI는 include directive를 사용해서 다른 SSI 자원을 포함할 수 있는데, include된 SSI 자원이 또 다시

include를 사용할 때 loop이 있을 가능성이 있으므로 중첩된 SSI 자원의 include depth를 적절히 설정하여 무한루프에 의한 서버 자원 낭비를 예방할 수 있다.

ForceCacheModificationCheck

- 종류: Boolean
- 기본값: N
- Conditional-GET 요청이 아닐 경우에도 항상 캐시하고 있던 응답이 유효한지 체크할지 여부를 설정한다(SVRTYPE이 HTML인 경우).
- HTH 캐시는 Conditional-GET 요청이 있을 경우에만 캐시하고 있던 응답이 유효한지 체크한다.

Y로 설정할 경우 모든 요청에 대해 항상 캐시하고 있던 응답이 유효한지 체크하면서 stat() system call을 해야 하므로 성능이 저하될 수 있다.

DebugHTHMemory

- 종류: Boolean
- 기본값: N
- HTH의 메모리 사용량을 모니터링할 경우 Y로 설정한다.

"wsadmin -C hthmem" 명령을 사용하기 위해서는 반드시 Y로 설정해야 한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

DOSBlock

- 종류: Boolean
- 기본값: N
- DoS 공격을 차단하는 설정을 사용하는 경우 Y로 설정한다.

DOSBlockTableSize

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 3097
- DoS 공격 차단 기능에서 내부적으로 사용하는 Hash 테이블의 크기를 설정한다.

DOSBlockPageCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 5
- DOSBlockPageInterval 설정시간 동안 같은 페이지를 설정된 값보다 더 많이 요청한 사용자 IP를 DOSBlockPeriod 동안 차단한다.
- 0으로 설정하는 경우 같은 페이지에 대한 DoS 공격을 체크하지 않는다.

DOSBlockPageInterval

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 같은 페이지에 대한 DoS 공격을 판단하는 주기를 설정한다.

DOSBlockSiteCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 50
- DOSBlockSiteInterval 설정시간 동안 사이트에 설정된 값보다 더 많이 요청한 사용자 IP를 DOSBlockPeriod 동안 차단한다.
- 0으로 설정하는 경우 사이트 요청에 대한 DoS 공격을 체크하지 않는다.

DOSBlockSiteInterval

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX

- 기본값: 1
- 사이트에 대한 DoS 공격을 판단하는 주기를 설정한다.

DOSBlockPeriod

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX
- 기본값: 30
- DoS 공격으로 판단한 IP를 차단할 시간을 설정한다.

DOSBlockWhiteList

- 종류: Literal
- 범위: 255자 이내
- DoS 공격 차단에서 제외한 IP를 설정한다. 앞단에 Proxy 서버를 사용할 경우 해당 서버의 IP를 설정한다.
- 콤마(,)를 사용하여 여러 IP를 설정할 수 있으며, 설정할 IP가 많을 경우 여러 항목으로 나누어 설정할 수 있다.
- 설정 IP는 총 256개까지 설정할 수 있다.

BindIPv6Only

- 종류: Boolean
- 기본값: <OS default>
- IPv6 사용하는 경우 bind하는 주소의 범위를 설정한다. 설정하지 않았을 경우 OS의 기본 설정을 사용한다.

설정값	설명
Y	IPv6를 사용하는 경우 IPv6 주소만 bind한다.
N	IPv6를 사용하는 경우 dual-stack으로 IPv4와 IPv6를 모두 같이 사용하도록 bind한다.

JsvListen

- 종류: Literal
- 범위: 1023자 이내
- 특정 IP에만 JSV 서비스 포트를 열고 싶을 경우 "<IP>:<Port>"로 설정한다.
- 콤마(,)로 구분하여 10개까지 설정할 수 있다.
- JsvPort 설정보다 우선되며, JsvListen을 설정한 경우 JsvPort 설정은 무시된다.

JsvSslListen

- 종류: Literal
- 범위: 1023자 이내
- 특정 IP에만 JSVSsl 서비스 포트를 열고 싶을 경우 "<IP>:<Port>"로 설정한다.
- 콤마(,)로 구분하여 10개까지 설정할 수 있다.
- JsvSslPort 설정보다 우선되며, JsvSslListen을 설정한 경우 JsvSslPort 설정은 무시된다.

UpperDirRestrict

- 종류: Boolean
- 기본값: N
- Y로 설정할 경우 Request URI에 상위 디렉터리로 접근하려는 '../'가 포함되어 있으면 "403 Forbidden"으로 응답하도록 한다.

CheckPingTimeoutStatus

- 종류: Numeric
- 범위: 511~599
- 기본값: 503
- JSV 서버로 PING을 보내고 이에 대한 응답을 기다리던 중 TimeOut이 발생할 경우 해당 설정의 값으로 HTTP Status Code를 설정하여 회신한다.

IgnoreMissingColonErr

- 종류: Boolean
- 기본값: N
- HTTP Request header가 "name:value" 형식을 지키지 않을 경우 에러 처리를 할지 여부를 결정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	설정할 경우 해당 header는 무시한다.
N	400 Bad Request 응답을 전송한다.

HTTP2Flag

- 종류: Boolean
- 기본값: N
- Y로 설정할 경우 HTTP/2 Connection 및 Request를 허용한다.
- SSL/TLS 프로토콜을 사용하는 경우에만 적용되기 때문에 HTTP/2를 사용하려는 NODE 절 또는 VHOST 절에는 SSLFlag를 설정해주어야 한다.

3.3.2. 예제

다음은 NODE 절을 설정한 예제이다.

```
*NODE
mynode
    WebtoBDir = "$WEBTOBDIR", # 환경변수 WEBTOBDIR을 참조
    SHMKEY = 0x8000,          # 0x를 앞에 붙여서 16진수로 설정 가능
    Docroot="docs/",          # "$WEBTOBDIR/docs/"와 동일한 의미
    Port = "8080",            # 주의! Numeric이 아닌 Literal type임
    JsvPort = 9900,
    HTH = 1,
    Logging = "access_log",
    ErrorLog = "error_log",
    SysLog = "system_log"
```

3.4. VHOST 절

Virtual Hosting이 필요한 경우 VHOST 절에 관련 환경을 설정한다.

Virtual Host기능은 실제로는 하나의 WebtoB가 동작하지만 각기 다른 URL로 다른 문서를 제공하도록 하므로써 마치 여러 개의 서버가 서비스를 제공하는 것처럼 보이도록 하는 기능이다.

VHOST 절에는 다음과 같은 내용이 정의된다.

- Virtual Host에 해당하는 노드 이름
- Virtual Host에 해당하는 호스트 이름
- 각 Virtual Host가 서비스할 HTML 문서들이 들어있는 최상위 디렉터리

또한 NODE 절에서 정의한 다음 내용은 재정의할 수 있다.

- 각종 로그 메시지 디렉터리 경로명
- 디렉터리 아이콘 경로명과 인덱싱 방식

NODE 절에서 이미 정의된 값들도 VHOST 절에서 재정의된 값으로 수정된다. Method, Expires, UsrLogDir, EnvFile, ErrorDocument, Logging Option 항목은 NODE 절에서 정의된 것과 같은 역할을 한다.

3.4.1. 설정 항목

다음은 VHOST 절의 환경설정 형식이다.

```
*VHOST
name    # String[32]
    Hostname = Literal[128],
   #HostAlias = Literal[1024],          # LIST
   #DocRoot = Literal[256],             # <*NODE's>, $ENV, R.PATH
   #Method = Literal[256],              # "GET,POST,HEAD,OPTIONS", PM.LIST
   #Port = Literal[1024],               # "80" or "443", 1 ~ 65535, LIST[100]
   #Listen = Literal[1024],             # LIST[100]
   #SslFlag = Boolean,                  # N
   #SslName = String[32],
   #IconDir = Literal[256],             # $ENV, R.PATH
   #UserDir = Literal[256],             # MULTI
   #EnvFile = String[256],              # $ENV, R.PATH
   #IndexName = Literal[256],           # "index.html", LIST
   #DirIndex = Literal[32],             # LIST
   #Options = Literal[256],             # "HTML,CGI,SSI,PHP,JSV", PM.LIST
   #ErrorDocument = Literal[256],       # LIST[64]
   #Logging = Literal[256],             # LIST[4]
   #ErrorLog = Literal[256],            # LIST
   #Filter = Literal[256],              # LIST[64]
   #DefaultMimetype = Literal[128],
   #Expires = Literal[256],             # LIST[64]
   #ServiceOrder = Literal[256],        # <*NODE's>
   #Keepalive = Boolean,                # <*NODE's>
   #KeepaliveTimeout = Numeric,         # <*NODE's> (1-)
   #KeepaliveMax = Numeric,             # <*NODE's> 0 (0-)
   #Timeout = Numeric,                  # <*NODE's> (1-)
   #DefaultCharset = Literal[32],
   #URLRewrite = Boolean,               # N
   #URLRewriteConfig = Literal[256],    # R.PATH
   #Headers = Literal[256]              # LIST[15]
   #CheckURL = Boolean,                 # N
   #CheckURLFrom = Literal[32],         # "utf-8"
   #CheckURLTo = Literal[32]
   #CheckUrlJsvExcept = Boolean,        # N
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

VHOST 절 이름

- 종류: String
- 범위: 31자 이내
- 정의할 VHOST 절 이름을 설정한다.
- VHOST는 64개까지 설정할 수 있다.

Hostname(필수 항목)

- 종류: Literal

- 범위: 127자 이내
- Virtual Host에 접근할 때 사용자가 사용할 호스트 이름을 적어준다.
- Name-based Virtual Hosting을 하는 경우 각각의 VHOST 절에서 호스트 이름을 다르게 설정하여 호스트 이름으로 Virtual Host가 구분한다.

HostAlias

- 종류: Literal
- 범위: 1023자 이내
- Hostname 항목에 설정된 호스트 이외에 다른 호스트를 추가하는 경우 HostAlias로 등록하여 설정한다.
- 콤마(,)로 구분하여 여러 개를 설정할 수 있으며, 애스터리스크(*) 및 물음표(?)를 사용할 수 있다. "*"는 모든 호스트를 의미한다.

DocRoot

- 종류: Literal
- 범위: 255자 이내
- 정의된 Virtual Host가 서비스하게 될 HTML 문서가 있는 최상위 디렉터리 경로명을 설정한다. 설정하지 않으면, NODE 절의 DocRoot를 사용한다.
- 다음 지시자를 사용해서 디렉터리 설정 패턴을 동적으로 설정할 수 있다.

지시자	설명
%p	요청한 포트 번호로 치환한다.
%n	Hostname이나 IP 주소의 n번째 요소로 치환한다. 만약 n을 0으로 하면 전체 문자열이 사용된다. 마이너스 기호(-)가 앞에 오면 Hostname이나 IP 주소의 끝에서부터 센다. 플러스 기호(+)가 뒤에 오면 Hostname이나 IP 주소의 나머지가 사용된다.
%n.m	n번째 요소의 m번째 문자로 치환한다. 위와 같이 마이너스 기호(-)나 플러스 기호(+)가 붙을 수 있다.
%%	단일 퍼센트(%) 표시로 치환한다.

지시자를 사용할 경우 DocRoot를 절대 경로로 설정해야 한다. 환경변수를 사용했을 경우 치환된 경로가 최종적으로 절대 경로일 경우에만 위 지시자를 사용할 수 있다.

Method

- 종류: Literal
- 범위: 255자 이내

- 기본값: "GET,POST,HEAD,OPTIONS"
- 처리할 수 있는 HTTP Request 메소드를 설정한다. 클라이언트로부터 받은 Request 메소드가 설정되어 있지않으면, 해당 요청은 처리되지 않는다.
- GET, POST, HEAD, OPTIONS, CONNECT, TRACE, PROPFIND, PUT, DELETE, MKCOL, COPY, MOVE를 지원한다.
- 특정 메소드를 서비스하지 않을 경우 "–OPTIONS" 와 같이 하이픈(-)을 메소드 이름 앞에 설정한다.

Port

- 종류: Literal
- 범위: 1023자 이내, 1 ~ 65535
- 기본값: "80" 또는 "443" (SSLFlag를 'Y'로 설정한 경우에는 "443"을 기본값으로 사용한다.)
- 사용자가 접속할 수 있는 서비스 포트를 설정한다.
- 콤마(,)로 구분하여 100개까지 설정 가능하다.

Port와 Listen 항목이 같이 설정된 경우 Port 설정은 무시된다.

Listen

- 종류: Literal
- 범위: 1023자 이내
- 특정 IP에만 서비스 포트를 열고 싶을 경우 "<IP>:<Port>"로 설정한다.
- 콤마(,)로 구분하여 100개까지 설정 가능하다.
- Port 항목에 설정된 값보다 우선되며, Listen을 설정한 경우 Port 설정은 무시된다.

SSLFlag

- 종류: Boolean
- 기본값: N
- 서비스 포트를 SSL/TLS 프로토콜을 사용하여 서비스할지 여부를 설정한다.
- SSLName 설정으로 적용할 SSL 절 항목을 지정할 수 있다.

SSLName

- 종류: String
- 범위: 31자 이내

- SSLFlag를 'Y'로 설정하는 경우 적용되며, 사용할 SSL 절 항목을 설정한다.

IconDir

- 종류: Literal
- 범위: 255자 이내
- 디렉터리 인덱싱 기능에서 사용하는 아이콘이 위치한 경로를 설정한다.
- 해당 설정값은 html tag에서 사용하는 경로가 되기 때문에 슬래시(/)와 DocRoot를 기준으로 하는 상대 경로로 설정해야 한다.

UserDir

- 종류: Literal
- 범위: 255자 이내
- 사용자별 디렉터리를 설정하는 경우 사용한다.
- 다음의 3가지로 설정할 수 있다.
  - <*per-user-home-directory*>
  - disabled [<*user-list*>]
  - enable [<*user-list*>]

EnvFile

- 종류: String
- 범위: 255자 이내
- 서버 프로세스를 기동하기 전에 환경변수를 전달할 경우 환경변수를 설정한 파일의 경로를 설정한다.
- SVRGROUP 절에 EnvFile 항목이 설정된 경우 VHOST 절 설정은 무시되며, SVRGROUP 절 설정만 사용한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

NODE 절의 EnvFile 항목 설정은 WebtoB 전체에 영향을 주지만, VHOST 및 SVRGROUP 절 설정은 서버 프로세스에만 적용된다.

IndexName

- 종류: Literal
- 범위: 255자 이내

- 기본값: "index.html"
- 디렉터리에 대한 요청을 하는 경우 기본으로 처리할 문서의 이름을 설정한다.

DirIndex

- 종류: Literal
- 범위: 31자 이내
- 적용할 DIRINDEX 절의 이름을 설정한다.

디렉터리 인덱싱 기능을 사용하기 위해서는 Options 항목 설정에 "INDEX"를 추가해야 하며, DIRINDEX 절 설정을 통해 디렉터리 인덱싱 기능의 세부 동작 방식을 변경할 수 있다.

Options

- 종류: Literal
- 범위: 255자 이내
- 기본값: "HTML,CGI,SSI,PHP,JSV,USER"
- VHOST가 제공하는 서비스와 기타 작동방식에 대해 설정한다.
- 특정 옵션을 사용하지 않을 경우 옵션 이름 앞에 하이픈(-)을 설정한다.
- 다음은 설정 가능한 옵션에 대한 설명이다.

옵션	설명
HTML	HTML 서비스를 처리한다.
CGI	CGI 서버를 사용한다.
PHP	PHP 서버를 사용한다.
SSI	SSI 서버를 사용한다.
JSV	JSV (JEUS) 서버를 사용한다.
INDEX	클라이언트가 특정 파일 이름을 지정하지 않고 디렉터리에 요구를 보낼 때 디렉터리의 내용을 보여준다. 예를 들어 "/docroot/dir/" 요청이 오는 경우 해당 디렉터리에 존재하는 파일 이름과 정보가 클라이언트로 전달된다.
ALL	"HTML, CGI, PHP, SSI, JSV, INDEX"와 같다.

ErrorDocument

- 종류: Literal
- 범위: 255자 이내
- HTTP 에러 페이지를 사용자가 지정한 페이지로 대신 사용할 경우 ERRORDOCUMENT 절에 정의한 이름을

설정한다.

- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

Logging

- 종류: Literal
- 범위: 255자 이내
- 기본값: <NODE 절 설정>
- 액세스 로그에 해당되는 LOGGING 절 이름을 설정한다.
- 콤마(,)로 구분하여 4개까지 설정할 수 있다.

ErrorLog

- 종류: Literal
- 범위: 255자 이내
- 기본값: <NODE 절 설정>
- 에러 로그에 해당되는 LOGGING 절 이름을 설정한다.

Filter

- 종류: Literal
- 범위: 255자 이내
- 사용할 FILTER 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

DefaultMimetype

- 종류: Literal
- 범위: 127자 이내
- 기본값: <NODE 절 설정>
- MIME-Type을 결정할 수 없는 문서의 Default Content-Type을 설정한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultMimetype을 사용한다. 둘 다 설정되지 않았을 경우 VHOST 및 NODE 절 설정을 사용한다.

Expires

- 종류: Literal
- 범위: 255자 이내
- 적용할 EXPIRES 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

ServiceOrder

- 종류: Literal
- 범위: 255자 이내
- 기본값: <NODE 절 설정>
- 사용자 요청을 처리할 서버와 서비스를 결정할 때 URI 절과 EXT 절의 우선순위를 결정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
uri,ext	URI 절 설정을 먼저 확인하고, 해당되는 설정이 없는 경우 EXT 절 설정을 확인한다.
ext,uri	EXT 절 설정을 먼저 확인하고, 해당되는 설정이 없는 경우 URI 절 설정을 확인한다.

URI 절과 EXT 절 모두 해당되지 않을 경우 기본 HTML 서비스에서 해당 요청을 처리한다.

Keepalive

- 종류: Boolean
- 기본값: <NODE 절 설정>
- Keepalive(HTTP persistent connection) 사용 여부를 결정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	사용자는 연결을 재사용하여 한 번의 연결로 여러 개의 요청을 처리할 수 있다.
N	사용자는 요청을 처리하기 위해 매번 소켓을 다시 연결해야 한다.

KeepaliveTimeout

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX

- 기본값: <NODE 절 설정>
- Keepalive를 유지하는 시간을 설정한다.
- 사용자 요청 처리가 끝난 후 설정된 시간이 지나면 연결을 끊는다.

KeepaliveMax

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: <NODE 절 설정>
- Keepalive 요청 건수를 제한하는 경우 설정한다.
- 0으로 설정하는 경우 요청 건수를 제한하지 않는다.

Timeout

- 종류: Numeric
- 단위: 초
- 범위: 1 ~ INT_MAX
- 기본값: <NODE 절 설정>
- 사용자가 연결한 소켓에서 데이터를 읽거나 쓸 때 적용하는 타임아웃을 설정한다.
- 사용자 요청을 처리하고 있을 때 적용되는 설정이며, 해당 시간 동안 사용자가 소켓에 데이터를 쓰지 않거나 소켓으로부터 데이터를 읽지 않는 경우 해당 소켓을 닫는다.

DefaultCharset

- 종류: Literal
- 범위: 32자 이내
- HTTP Response Header 중 "Content-Type" 항목에 추가할 "charset=<value>"의 기본값을 정의한다.

설정값	설명
On	"ISO-8859-1"을 기본값으로 사용한다.
Off	charset을 추가하지 않는다.
<custom-charset>	설정한 "<custom-charset>"을 기본값으로 사용한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultCharset을 사용한다. 둘 다 설정되지 않은 경우 VHOST 절 및 NODE 절 설정을 사용한다.

URLRewrite

- 종류: Boolean
- 기본값: N
- URLRewrite 기능을 사용할지 여부를 결정한다.

이 기능을 사용하려면 URLRewriteConfig에 설정 파일의 경로를 지정해야 한다.

URLRewriteConfig

- 종류: Literal
- 범위: 255자 이내
- URLRewirte 기능을 사용하기 위한 설정 파일의 경로를 지정한다. 자세한 설정은 URLRewrite를 참고한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

CheckURL

- 종류: Boolean
- 기본값: N
- HTTP Request URL path의 Charset을 변환할지 여부를 결정한다.
- Charset은 CCS(Coded Character Set)와 CES(Character Encoding Scheme)의 조합으로, 한글은 일반적으로 EUC-KR이나 UTF-8을 사용한다.
- Request URL path에 사용한 Charset과 서버에서 사용하는 Charset이 다를 경우 CheckURL을 사용해서 Charset을 변환하면 된다.
- Y로 설정한 경우 CheckURLFrom과 CheckURLTo를 설정해야 한다.
- NODE 절과 함께 설정한 경우 VHOST 절의 설정을 우선한다.

"euc-kr"과 "utf-8"만 지원한다.

CheckURLFrom

- 종류: Literal
- 범위: 31자 이내
- 기본값: "utf-8"
- HTTP Request URL path가 사용하는 Charset을 설정한다.

"euc-kr"과 "utf-8"만 지원한다.

CheckURLTo

- 종류: Literal
- 범위: 31자 이내
- 서버가 사용하는 Charset을 설정한다.

CheckUrlJsvExcept

- 종류: Boolean
- 기본값: N
- JEUS로 포워딩하는 요청에 대하여 CheckURL 설정을 적용하지 않을지 여부를 결정한다.
- Y로 설정한 경우 JEUS로 요청을 보낼 때에는 URI를 변환하지 않는다.

3.4.2. 예제

다음은 VHOST 절을 설정한 예제이다.

```
*VHOST
webtob
    Docroot = "docs/vhost_docs",
    Hostname = "webtob.tmax.co.kr",
    Port="80",
    IndexName = "welcome.html",
    Logging = "webtob_access",
    ErrorLog = "webtob_error"
```

3.5. HTH_THREAD 절

HTH_THREAD 절은 HTH 프로세스 내 Thread를 설정한다. HTH_THREAD 절은 반드시 설정해야 하며 하나만 설정한다.

3.5.1. 설정 항목

다음은 HTH_THREAD 절의 환경설정 형식이다.

```
*HTH_THREAD
name    # String[32]
        WorkerThreads = Numeric            # 8 (1-100)
       #ReadBufSize = Numeric,             # 1048576 (65536-)
       #SendfileThreads = Numeric,         # 4 (0-100)
       #SendfileThreshold = Numeric,       # 0 (0-)
       #AccessLogThread = Boolean,         # Y
       #HtmlsCompression = Literal[256],   # LIST[32], MULTI
       #HtmlsCompressionMinSize = Numeric, # 1 (1-),
       #Schedule = Literal[256]            #"FA"
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

HTH_THREAD 절 이름

- 종류: String
- 범위: 31자 이내
- HTH_THREAD 이름을 설정한다.

WorkerThreads

- 종류: Numeric
- 범위: 1 ~ 100
- 기본값: 8
- Worker Thread의 수를 설정한다.
- Worker Thread는 정적 파일 처리, SSL/TLS 처리(Handshake, 암호화, 복호화), 압축 처리, HTTP 인증 처리를 하는 Thread이다.

ReadBufSize

- 종류: Numeric
- 단위: bytes
- 범위: 65536 ~ INT_MAX
- 기본값: 1048576
- Worker Thread에서 정적 파일을 읽기 위한 read buffer 크기를 설정한다.

SendfileThreads

- 종류: Numeric
- 범위: 0 ~ 100
- 기본값: 4
- Sendfile Thread의 수를 설정한다.
- 정적 파일을 처리할 때 특정 OS에서만 지원하는 sendfile 기능을 사용하는 Thread이며, blocking으로 동작한다.

이 기능은 UNIX/Linux 환경에서만 사용할 수 있다.

SendFileThreshold

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 요청한 파일의 크기가 설정된 값보다 크면 sendfile을 사용한다.
- 0으로 설정하는 경우 sendfile 기능을 사용하지 않는다.

이 기능은 UNIX/Linux 환경에서만 사용할 수 있다.

AccessLogThread

- 종류: Boolean
- 기본값: Y
- HTH에서 사용자 요청 처리후 접근 기록을 남기기 위한 Thread 사용 여부를 설정한다.
- 이 기능을 사용하지 않는 경우 WSM에서 접근 기록을 남기게 된다.

이 기능은 UNIX/Linux 환경에서만 사용할 수 있다.

HtmlsCompression

- 종류: Literal
- 범위: 255자 이내
- Worker Thread에서 처리하는 정적 파일에 대한 응답에 대해 압축할 대상을 결정하는 설정이다.

- 압축할 MIME-Type을 설정하며, 응답의 Content-Type이 사용된다. 해당되는 응답은 클라이언트로 전송되기 전에 Worker Thread에서 GZIP을 사용해서 압축된다.
- 콤마(,)로 구분하여 최대 32개까지 여러 MIME-Type을 지정할 수 있다.
- 압축기능을 사용할 경우 네트워크 트래픽을 줄일 수 있지만, 서버의 성능은 저하될 수 있다.

  압축률이 낮은 파일(zip 같은 압축 파일이나 jpeg 같은 압축 이미지 등)을 압축할 경우 서버에 부하만 주기 때문에 해당 MIME-Type이 압축 대상이 되지 않도록 주의해야 한다.

압축 기능은 HTTP Request Header 중 Accept-Encoding에 GZIP이나 deflate로 지정된 요청에 대해서만 적용된다.

HtmlsCompressionMinSize

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 압축 설정을 통해 응답을 압축하는 경우 응답의 최소 크기를 설정한다.
- 요청한 파일의 크기가 설정된 값보다 크면 응답을 압축한다.

Schedule

- 종류: Literal
- 범위: RR | FA
- 기본값: FA
- Worker Thread를 여러 개 설정하였을 경우 요청을 처리할 Thread를 지정하는 방법을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
RR	Round Robin 방식으로 순차적으로 Worker Thread에 요청을 할당한다.
FA	First Assign 방식으로 현재 가장 적은 일을 처리하고 있는 Worker Thread에 요청을 할당한다.

3.5.2. 예제

다음은 HTH_THREAD 절을 설정한 예제이다.

```
*HTH_THREAD
hworker
  WorkerThreads = 8,
  SendfileThreads = 4,
  SendfileThreshold = 32768,
```

```
  AccessLogThread = Y
```

3.6. SVRGROUP 절

WebtoB를 통해 응용 서버 프로세스를 접근하는 경우 서버 프로세스의 논리적인 연관성에 따라 이들을 그룹으로 관리할 필요가 있다. SVRGROUP 절에서는 이러한 그룹에 대한 환경설정을 한다.

SVRGROUP 절에는 다음과 같은 내용이 정의된다.

- 서버 그룹이 속한 호스트 이름
- 서버 그룹이 제공하는 서비스 타입

이 밖에 NODE 절이나 VHOST 절에서 정의한 내용이 서버 그룹에 따라 재정의할 수 있으며, 데이터베이스를 사용하는 경우 데이터베이스 접근과 관련된 정보들이 정의될 수 있다.

3.6.1. 설정 항목

다음은 SVRGROUP 절의 환경설정 형식이다.

```
*SVRGROUP
name    # String[32]
    SvrType = String[32],
   #VhostName = String[1024],           # LIST[64], MULTI
   #AppDir = Literal[256],              # <*NODE/VHOST's>, $ENV, R.PATH
   #UsrLogDir = Literal[256],           # $ENV, R.PATH
   #EnvFile = String[256],              # $ENV, R.PATH
   #AuthentName = String[32],
   #Logging = Literal[256],             # LIST[4]
   #ScriptLoc = Literal[256],
   #ScriptArgs = Literal[256],
   #Filter = Literal[256],              # LIST[64]
   #DefaultMimetype = Literal[128],
   #Expires = Literal[256],             # LIST[64]
   #DefaultCharset = Literal[32],
   #LBServers = Literal[256],           # LIST[32]
   #LBType = String[16],                # Dynamic, Dynamic | Static
   #LBBackup = Literal[32],
   #Headers = Literal[256],             # LIST[15]
   #UserAgentRegExp = String[512]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

SVRGROUP 절 이름

- 종류: String
- 범위: 31자 이내

- 서버 그룹의 이름을 설정한다.
- 서버 그룹에 대한 논리적인 이름으로써 SVRGROUP 절 내에서 유일(unique)한 값이어야 한다.

SvrType(필수 항목)

- 종류: String
- 범위: 31자 이내
- 서버 그룹에서 서비스할 서버 타입을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
HTML	정적 파일 요청을 서비스하는 서버 타입이다.
CGI	CGI 요청을 서비스하는 서버 타입이다.
PHP	PHP 요청을 서비스하는 서버 타입이다.
SSI	SSI 요청을 서비스하는 서버 타입이다.
JSV	JEUS와 연동하여 JSP, servlet 등의 요청을 서비스하는 서버 타입이다.
WEBSTD	WBAPI로 작성된 서버의 요청을 서비스하는 서버 타입이다.

VhostName

- 종류: Literal
- 범위: 1023자 이내
- 서비스할 VHOST 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정 가능하다.

AppDir

- 종류: Literal
- 범위: 255자 이내
- 기본값: <해당하는 NODE 혹은 VHOST 절 설정>
- WBAPI로 작성된 사용자 애플리케이션이 위치한 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정하는 경우 $WEBTOBDIR로부터의 절대 경로로 변환하여 설정된다.

UsrLogDir

- 종류: Literal
- 범위: 255자 이내
- 사용자 로그를 기록할 경로를 설정한다.
- Clopt 설정의 "-e <*stderr-log-file*> -o <*stdout-log-file*>" 옵션과 관련이 있으며, 설정된 경로에 해당 이름으로 로그를 남긴다.

UsrLogDir은 SERVER, SVRGROUP, NODE 절 순으로 먼저 설정된 값을 사용한다.

EnvFile

- 종류: String
- 범위: 255자 이내
- 서버 프로세스를 기동하기 전에 환경변수를 전달할 경우 환경변수를 설정한 파일의 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

NODE 절의 EnvFile 설정은 WebtoB 전체에 영향을 주지만, VHOST 및 SVRGROUP 절 설정은 서버 프로세스에만 적용된다.

AuthentName

- 종류: String
- 범위: 31자 이내
- 적용할 AUTHENT 절 이름을 설정한다.

AuthentName 설정은 DIRECTORY, URI, EXT, SVRGROUP 절에 모두 설정 가능하며, 순서대로 먼저 설정된 절의 값이 적용된다.

Logging

- 종류: Literal
- 범위: 255자 이내
- 액세스 로그에 해당되는 LOGGING 절 이름을 설정한다.
- 콤마(,)로 구분하여 4개까지 설정할 수 있다.

ScriptLoc

- 종류: Literal
- 범위: 255자 이내
- PHP에 관련된 서버 그룹을 설정할 경우 PHP 실행 모듈이 실제로 존재하는 곳의 경로를 설정한다.
- $WEBTOBDIR을 기준으로 상대 경로를 설정한다.

PHP 실행 모듈 중 "php-cgi"에 해당되는 경로를 설정하고, UNIX/Linux 환경에서는 symbolic link를 사용하여 $WEBTOBDIR 아래 경로를 사용한다.

ScriptArgs

- 종류: Literal
- 범위: 255자 이내
- PHP에 관련된 서버 그룹을 설정할 경우 PHP 실행 모듈에 대한 파라미터를 설정한다.
- 설정된 값을 그대로 파라미터로 적용하므로 옵션에 경로가 들어간다면 절대 경로를 사용해야 한다.

Filter

- 종류: Literal
- 범위: 255자 이내
- 사용할 FILTER 절 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

DefaultMimetype

- 종류: Literal
- 범위: 127자 이내
- MIME-Type을 결정할 수 없는 문서의 Default Content-Type을 설정한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultMimetype을 사용한다. 둘 다 설정되지 않았을 경우 VHOST 및 NODE 절 설정을 사용한다.

Expires

- 종류: Literal
- 범위: 255자 이내
- 적용할 EXPIRES 절 이름을 설정한다.

- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

DefaultCharset

- 종류: Literal
- 범위: 32자 이내
- HTTP Response Header 중 "Content-Type" 항목에 추가할 "charset=<value>"의 기본값을 정의한다.

설정 값	설명
On	"ISO-8859-1"을 기본값으로 사용한다.
Off	Charset을 추가하지 않는다.
<custom-charset>	설정한 "<custom-charset>"을 기본값으로 사용한다.

DIRECTORY 절에 설정된 경우 우선 사용하며, 다음으로 SVRGROUP 절에 설정된 DefaultCharset을 사용한다. 둘 다 설정되지 않았을 경우 VHOST 및 NODE 절 설정을 사용한다.

LBServers

- 종류: Literal
- 범위: 255자 이내
- 서버들 사이에 부하 분산을 설정하고자 할 경우 원하는 서버의 목록을 설정한다.
- 콤마(,)로 구분하여 32개까지 설정할 수 있다.

LBType

- 종류: String
- 범위: 15자 이내
- 기본값: Dynamic
- LBServers를 설정한 경우 서버들 간의 부하를 분배하는 방법을 설정한다.

설정 값	설명
Dynamic	기본적으로 Round Robin으로 분배하지만, 서버 큐에 대기 중인 요청의 수가 해당 서버의 현재 서버 프로세스 수 이상인 경우는 제외하고 분배한다. 만약 모든 서버의 서버 큐에 요청이 쌓여 있는 경우에는 다시 Round Robin으로 분배한다.
Static	SERVER 절의 LBFactor에 비례하여 서버가 요청을 처리하도록 분배한다.

LBBackup

- 종류: Literal
- 범위: 31자 이내
- LBServers로 설정한 서버들이 모두 READY 상태가 아닐 경우 백업 용도로 사용할 서버를 설정한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

UserAgentRegExp

- 종류: String
- 범위: 511자 이내
- USER-AGENT에 따라 요청을 보낼 JEUS 서버를 설정한다.
- 해당 Agent로부터 들어온 요청은 해당 서버 그룹에 포함된 서버가 처리한다.

3.6.2. 예제

다음은 SVRGROUP 절을 설정한 예제이다.

```
*SVRGROUP
htmlg
    SvrType = HTML
phpg
    SvrTYpe = PHP,
    ScriptLoc = "bin/php-cgi"
    ScriptArgs = "-c /php_conf/php.ini"
cgig
    SvrType = CGI
ssig
    SvrType = SSI
jsvg
    SvrType = JSV,
    LBServers = "jsv1, jsv2, jsv3",
    LBType = Static,
    LBBackup = "jsv4"
wbapg
    SvrType = WEBSTD
```

3.7. SERVER 절

SERVER 절에서는 실질적으로 제공하는 서비스들을 등록한다. WebtoB는 등록된 서비스만을 처리하기 때문에 새로운 서버 프로그램이 추가되는 경우 서버의 환경 파일에 반드시 설정해야 한다.

WebtoB가 제공하는 대부분의 서비스는 SERVER 절에서 등록이 가능하며 비즈니스 로직을 WebtoB를 통해 직접 호출하는 경우에만 SERVICE 절의 설정이 필요하다. 각각의 서버는 위의 SVRGROUP 절에 정의된 서비스 종류에 따라 CGI, JSV 등으로 구분되며 서버 그룹 이름과 프로세스의 가능한 개수 등을 설정한다.

HTH의 Worker Thread에서 HTML 서비스를 모두 처리하므로 HTML을 위한 서버는 따로 설정하지 않는다.

3.7.1. 설정 항목

다음은 SERVER 절의 환경설정 형식이다.

```
*SERVER
name    # String[32]
    SvgName = String[32],
   #Clopt = Literal[256],
   #MinProc = Numeric,                  # 1 (1-)
   #MaxProc = Numeric,                  # <MinProc> (1-)
   #WSProc = Numeric,                   # 0 (0-)
   #UsrLogDir = Literal[256],           # $ENV, R.PATH
   #MaxQCount = Numeric,                # 0 (0-)
   #MaxQUrl = Literal[256],
   #MaxQUrlRedirectStatus = String[8],
   #ASQCount = Numeric,                 # 0 (0-)
   #FlowControl = Numeric,              # 50 (1-)
   #MaxRestart = Numeric,               # 20 (0-)
   #SvrCPC = Numeric,                   # 1 (1-)
   #SvrType = String[32],
   #HttpOutBufSize = Numeric,           # 8192 (0-)
   #HttpInBufSize = Numeric,            # 8192 (0-)
   #MaxRequests = Numeric,              # 0 (0-)
   #SvrChkTIme = Numeric,               # 0 (0-)
   #Schedule = String[256],             # "RR"
   #Options = Literal[256],             # PM.LIST
   #SessionIdCookieKey = Literal[256],
   #SessionIdUrlKey = Literal[256],
   #FlexibleStickySessionRouting = Boolean,  # N
   #Compression = Literal[256],         # LIST[32], MULTI
   #CompressionMinSize = Numeric,       # 1 (1-)
   #VhostName = String[1024],           # LIST[64], MULTI
   #FcgiInitEnv = Literal[256],         # MULTI[32]
   #FcgiKillTimeout = Numeric,          # 0 (0-)
   #FcgiKillMaxRequest = Numeric,       # 0 (0-)
   #Headers = Literal[256],             # LIST[15]
   #LBFactor = Numeric,                 # 1 (1-1000)
   #MaxJengineCount = Numeric,          # 32 (0-)
   #RequestLevelPing = Boolean,         # N
   #RequestLevelPingTimeout = Numeric,  # 3 (0-)
   #RequestLevelPingRetryCount = Numeric # 0 (0-)
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을

참고한다.

SERVER 절 이름

- 종류: String
- 범위: 31자 이내
- 하나의 서버를 의미하는 이름으로써 일반적으로 서버 이름은 유일(Unique)해야 한다. 하나의 서버 이름은 SERVER 절에 단 한 번만 정의되어야 한다. 같은 이름을 중복하여 이용하면 환경 파일의 컴파일될 때 에러가 발생하게 된다.

서버 타입이 JSV인 경우 서버 이름은 15자 이하를 사용해야 한다. 더 긴 이름을 설정할 경우 JEUS와 통신이 불가능하다.

SvgName(필수 항목)

- 종류: String
- 범위: 31자 이내
- 서버가 속해 있는 서버 그룹을 설정한다.
- 설정된 값은 반드시 SVRGROUP 절에서 정의된 서버 그룹 이름이어야 한다.
- 서버와 SVRGROUP 절의 연결을 통해서 서버가 어떤 노드에서 동작할 것인지, 어떤 리소스 매니저(데이터베이스)를 사용하는지 알 수 있다. 해당 리소스 매니저를 열 때 필요한 파라미터를 넘겨 줄 수 있다.

Clopt

- 종류: Literal
- 범위: 255자 이내
- 서버 프로세스가 기동될 때 그 서버 프로세스로 전달되는 명령어 옵션이 있을 경우 설정한다.
- 설정된 옵션들 중에 '--' 이전에 지정된 옵션들은 시스템에서 사용하고, 그 이후에 지정된 옵션들은 사용자가 자유롭게 사용할 수 있다.

MinProc

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 기본적으로 기동될 서버 프로세스의 개수를 설정한다.
- 기존의 클라이언트/서버 모델에서는 클라이언트당 서버 프로세스가 하나씩 기동되는 형식으로 동작하였으나

WebtoB는 그 보다 효율적인 구조를 가지고 있다. WebtoB에서는 서버 프로세스의 수는 일정하게 유지하고 하나의 서버 프로세스가 여러 개의 클라이언트 요구를 서비스할 수 있다.

- MinProc는 서버 프로세스의 개수를 조절하는 것으로써 운영 경험을 통해 적절한 개수를 지정할 필요가 있다. MinProc는 서버 프로세스의 최소 개수를 나타내는 것으로 처음 WebtoB가 기동될 때 시작되는 프로세스의 수와 같다.

MaxProc

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: <MinProc>
- MinProc를 포함하여 추가적으로 기동시킬 수 있는 프로세스의 최대 개수를 설정한다.
- 서버 프로세스는 기본적으로 WebtoB가 기동되는 시점에 설정된 MinProc 개수만큼 기동되고, 부하가 높아지는 경우 MaxProc 개수까지 서버 프로세스가 자동적으로 기동된다.
- Clopt 설정의 "-e <*stderr-log-file*> -o <*stdout-log-file*>" 옵션과 관련이 있으며, 설정된 경로에 해당 이름으로 로그를 남긴다.

WSProc

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값 : 0
- WebSocket 요청에 대해 WebtoB와 JEUS 간의 Reverse Connection 방식을 사용하기 위한 연결을 설정한다.
- JSV 타입의 서버인 경우에만 설정 가능하다.
- 기본값은 0이며 설정하지 않을 경우 Reverse Conncetion 방식으로 WebSocket을 사용할 수 없다. (Reverse Proxy 방식으로 동작 가능)
- WebSocket을 지원하는 JEUS의 ms 당 1개씩 설정하여 사용한다.

UsrLogDir

- 종류: Literal
- 범위: 255자 이내
- 사용자 로그를 기록할 경로를 설정한다.
- Clopt 설정의 "-e <*stderr-log-file*> -o <*stdout-log-file*>" 옵션과 관련이 있으며, 설정된 경로에 해당 이름으로 로그를 남긴다.

UsrLogDir은 SERVER, SVRGROUP, NODE 절 순서로 먼저 설정된 값을 사용한다.

MaxQCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 클라이언트의 요청이 엄청난 폭주를 이루어 정상적인 서비스 처리가 어려울 경우 계속되는 서비스 요청을 무시할 필요가 있다. 큐에 쌓인 클라이언트의 요구 수가 어느 정도 이상이 되면 새로 도착한 요구는 큐에 쌓이지 않고 클라이언트에 즉시 에러로 응답한다.
- MaxQCount는 큐의 쌓이는 요구 한계를 설정한다. MaxQCount에 설정한 값만큼 사용자 요구를 저장하여 기다리게 한다. 만약 이 값이 크다면 사용자의 요구를 문제 없이 처리할 수 있다는 장점이 있으나 너무 크게 하면 사용자에게 응답을 너무 늦게 반환할 수도 있다는 문제점이 있다.

MaxQUrl

- 종류: Literal
- 범위: 255자 이내
- 서버 큐가 다 찼을 경우 대신 서비스할 페이지를 설정한다.

MaxQUrlRedirectStatus

- 종류: String
- 범위: 7자 이내
- MaxQUrl 설정하는 경우 Redirect Status Code를 설정한다.
- 다음과 같은 값을 사용할 수 있다.

설정값	별칭	설명
301	permanent	"301 Moved Permanently"로 응답한다.
302	found	"302 Found"로 응답한다.
303	seeother	"303 See Other"로 응답한다.
305	useproxy	"305 Use Proxy"로 응답한다.
307	temp	"307 Temporary Redirect"로 응답한다.
410	gone	"410 Gone"으로 응답한다.

ASQCount

- 종류: Numeric

- 범위: 0 ~ INT_MAX
- 기본값: 0
- 자동으로 서버 프로세스를 추가 기동하기 위한 조건으로 큐에 쌓여진 요구 개수를 설정한다. 큐에 설정된 이상의 것이 쌓이게 되면 MinProc에서 MaxProc에 설정된 수만큼 차례대로 증가하게 된다.
- 0으로 설정된 경우 서버 프로세스의 수는 변동없이 MinProc에 설정된 수를 유지한다.

FlowControl

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 50
- HTH가 서버로부터 응답을 읽기위해 사용하는 버퍼의 크기를 설정할 때 Flow control하기 위한 단위를 설정한다. Flow control 버퍼의 크기는 아래와 같이 설정된다.

```
Flow control 버퍼 크기 = FlowControl * HttpOutBufSize
```

- 0으로 설정된 경우 기본값으로 적용된다.

MaxRestart

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 20
- 서버 프로세스의 최대 재시작 가능 횟수를 설정한다.

SvrCPC

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 특수한 서버 프로세스에서 HTH 프로세스와 병렬 통신 채널 수를 설정한다.
- 서버 프로세스의 처리량이 많아 하나의 채널로는 처리 속도가 저하될 때 병렬 통신으로 처리 속도를 증가시킬 수 있다.

HttpOutBufSize

- 종류: Numeric

- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 8192
- 서버가 사용자의 Request에 대한 응답을 보낼 때 사용하는 버퍼의 크기를 설정한다.
- 0으로 설정하거나 16777216(16MB)보다 크게 설정할 경우 서버 프로세스에서는 16777216(16MB)를 사용하고, HTH에서는 Flow control 버퍼를 만들기 위해서 기본값인 8192를 사용한다.

HttpInBufSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 8192
- 서버가 사용자의 Request를 받을 때 사용하는 버퍼의 크기를 설정한다.
- 0으로 설정하거나 16777216(16MB)보다 크게 설정할 경우 16777216(16MB)를 사용한다.

MaxRequests

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 무제한을 의미한다.)
- SERVER 절에 각 서버에 정의된 MaxRequests 값에 따라 각 서버의 프로세서들이 그 값만큼의 사용자 Request를 처리한 후 자동으로 재기동된다.
- WBAPI로 작성된 WEBSTD 서버인 경우 AP에 메모리 관련 버그가 있는 경우 유용하다.
- 서비스가 많은 경우 사용자 서비스의 연속성을 위해 MaxRequests보다 많은 Request를 처리한 후 재기동될 수도 있다.

SvrChkTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0 (JSV인 경우 60)
- 서버와의 연결이 정상인지 확인하기 위해 체크하는 시간을 설정하며 방화벽을 사이에 두고 연결된 JEUS와의 연결을 확인하는 데 사용된다.
- SvrChkTime을 설정하고 서비스 요청이 없는 Ready 상태의 커넥션에 대해서 연속된 2회의 SvrChkTime에

의한 KeepAlive 요청에도 응답이 없으면 해당 커넥션에 이상이 발생했다고 인식하고 해당 커넥션을 단절하여 서비스 분배에서 제외한다.

Schedule

- 종류: String
- 범위: RR | FA
- 기본값: RR
- 클라이언트로부터의 Request를 처리할 때 해당 Request를 받아서 처리할 서버 프로세스를 지정하는 방법을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
RR	Round Robin 방식으로 Idle한 서버 프로세스에 요청을 할당한다.
FA	First Assign 방식으로 우선순위(index)가 높고 Idle한 서버 프로세스에 요청을 할당한다.

Options

- 종류: Literal
- 범위: 255자 이내
- 해당 서버에 적용된 옵션을 설정한다.

옵션	설명
{+|-}Cache	Content의 캐시 유무(+|-)를 결정한다. SVRTYPE이 HTML, JSV인 서버인 경우 지원된다.
BlockListen	해당 서버가 서비스 불능 상태가 되면 더 이상의 클라이언트를 받아들이지 않도록 이 서버와 관련된 가상 호스트의 서비스 포트들을 닫는 기능이다. 서버의 상태가 복구되면 해당 포트들을 다시 서비스(listen)하여 새로운 클라이언트가 접속할 수 있도록 한다. L4 장비를 사용하여 여러 개의 노드로 구성된 경우 몇몇 노드가 서비스 불능 상태가 되었을 때 해당 노드의 Port Listen이 중단되므로 L4 장비는 서비스가 가능한 노드로 클라이언트를 분산시킬 수 있다. 동일한 서비스 포트를 사용하는 가상 호스트가 2개 이상인 경우 하나의 서비스 포트가 닫히게 되면 해당 서비스 포트와 관련된 가상 호스트 모두 서비스가 불가능해지므로 이 옵션을 사용하는 경우 가상 호스트별로 서비스 포트를 나눠서 사용해야 한다.

옵션	설명
BlockListenCloseClients	BlockListen 기능으로 서비스 포트들을 닫을 때 기존에 해당 포트들에 연결되어 있는 클라이언트들까지 닫고 싶을 경우 설정한다. 요청 처리 중인 클라이언트는 요청이 끝난 후 연결을 끊고, keepalive로 연결을 유지하고 있는 클라이언트는 즉시 연결을 끊는다.
NotifyClientClose	요청 처리 중 사용자가 연결을 끊었을 경우 서버에게 이를 알릴지 여부를 설정한다. JEUS의 경우 요청을 처리 중인 연결을 끊어서 JEUS에게 알려준다.
SecurityHolePassAuthorization	Authorization이나 Proxy-Authorization Header를 CGI/PHP의 환경변수로 전달할지 여부를 설정한다. 사용자 정보가 CGI/PHP에 노출될 수 있으므로 설정할 때 주의해야 한다.
503ResponseOnSuspend	**suspend** 명령으로 인해 해당 서버가 서비스 불능 상태가 되면 해당 서버가 처리해야 할 요청에 대해 "503 Service Temporarily Unavailable"로 응답하도록 하는 기능이다. 위와 같은 상황에서 이 옵션을 사용하지 않는 경우 해당 서버가 처리해야 할 요청들은 큐잉된다.
PassOriginalUriAfterFilters	FILTERS 프로세스에서 처리된 이후 요청을 JSV 서버(또는 다른 서버프로세스)에 요청을 전달할 때 사용자로부터 받은 요청을 그대로 전달할 때 사용하는 기능이다. FILTERS 프로세스에서 처리되면서 URL 인코딩 등으로 요청 URL이 변경될 경우를 고려하여 사용될 수 있다.
AllServers	HTMLS 요청에 대해 Filter 모듈을 사용 할 경우 Filter 기능을 FILTERS 프로세스에서 처리할지 여부를 결정한다.

SessionIdCookieKey

- 종류: Literal
- 범위: 255자 이내
- 기본값: "JSESSIONID"
- 서버 타입이 JSV인 경우 Session routing용으로 사용되는 HTTP Cookie의 Key 이름을 설정한다.

SessionIdUrlKey

- 종류: Literal
- 범위: 255자 이내
- 서버 타입이 JSV인 경우 Session routing용으로 사용되는 HTTP URL의 Key 이름을 설정한다.
- SessionIdCookieKey 항목이 설정된 경우 SessionIdCookieKey 설정을 우선한다.

FlexibleStickySessionRouting

- 종류: Boolean
- 기본값: N
- 서버 타입이 JSV인 경우 Session routing을 하게 되는데, 이때 flexible하게 routing할 것인지를 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	동일한 jengineid를 가진 서버 프로세스가 모두 RUN 상태인 경우 큐잉(Queuing)하지 않고 동일한 서버 내에서 다른 jengineid를 가진 서버 프로세스에 routing한다.
N(사용자가 설정하지 않은 경우)	기본으로 Sticky Session routing이 된다(이전 버전과 동일). 이는 동일한 jengineid를 가진 서버 프로세스에만 routing하고, 해당 서버 프로세스가 모두 RUN 상태이면 큐잉한다는 의미이다.

Flexible routing을 하는 경우 동일한 JSESSIONID를 가진 다른 클라이언트의 요청이 각각 다른 서버로 routing될 수 있으므로 주의해야 한다. 기본값 사용을 권장한다.

Compression

- 종류: Literal
- 범위: 255자 이내
- 응답을 압축할 대상을 결정하는 설정이다.
- 압축할 MIME-Type을 설정하며, 응답의 Content-Type이 사용된다. 해당되는 응답은 클라이언트로 전송되기 전에 GZIP을 사용해서 압축된다.
- 콤마(,)로 구분하여 최대 32개까지 여러 MIME-Type을 지정할 수 있다.
- 압축기능을 사용할 경우 네트워크 트래픽을 줄일 수 있지만, 서버의 성능은 저하될 수 있다.

  압축률이 낮은 파일(zip 같은 압축 파일이나 jpeg 같은 압축 이미지 등)을 압축할 경우 서버에 부하만 주기 때문에 해당 MIME-Type이 압축 대상이 되지 않도록 주의해야 한다.

압축 기능은 HTTP Request Header 중 Accept-Encoding에 GZIP이나 deflate로 지정된 요청에 대해서만 적용된다.

CompressionMinSize

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 압축 설정을 통해 응답을 압축하고자 할 때 응답의 최소 크기를 설정한다.

- Content-Length 응답 헤더의 값이 설정된 값보다 크면 응답을 압축한다. 단, chunked 응답인 경우 응답의 크기를 알기 어려우므로 적용되지 않는다.

VhostName

- 종류: Literal
- 범위: 1023자 이내
- 서버가 특정 Virtual Host 요청만을 처리할 경우 해당 VHOST 절의 Virtual Host 이름을 설정한다.
- SVRGROUP 절에 VhostName이 정의되어 있다면 서버의 VhostName과 SVRGROUP 절의 VhostName은 일치해야 한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

FcgiInitEnv

- 종류: Literal
- 범위: 255자 이내
- FastCGI 애플리케이션을 실행할 때 필요한 환경변수를 추가할 수 있다. 포맷은 'NAME=VALUE'이다.

```
FcgiInitEnv = "LOGFILE=/wb-413/log/myapp.log"
```

```
# myapp.log 예제
FAST_CGI_ENV_TEST="sample fast cgi environment variable"
```

- 1개 이상의 FcgiInitEnv를 사용해서 여러 개의 환경변수를 설정할 수 있다.

FcgiKillTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 무제한을 의미한다.)
- FastCGI의 요청을 처리한 후 애플리케이션 프로세스를 강제로 종료할지 여부를 설정한다.
- 애플리케이션 프로세스가 시작한 후 현재까지의 시간이 FcgiKillTimeOut에 지정한 시간보다 크면 서버는 요청을 요청을 처리한 후 애플리케이션 프로세스를 강제로 종료한다.

FcgiKillMaxRequest

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 무제한을 의미한다.)
- FastCGI의 요청을 처리한 후 애플리케이션 프로세스를 강제로 종료할지 여부를 설정한다.
- 애플리케이션 프로세스가 시작한 후 현재까지 처리한 요청 개수가 FcgiKillMaxRequest에 지정한 개수보다 크면 서버는 요청을 처리한 후 애플리케이션 프로세스를 강제로 종료한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

LBFactor

- 종류: Numeric
- 범위: 1 ~ 1000
- 기본값: 1
- SVRGROUP 절에 LBServers로 설정된 서버에 한하여 LBType이 Static인 경우 요청을 어떤 비율로 처리할지 설정한다.
- wsadmin에서 "set -v <server_name> lbf <value>" 명령을 통해 동적으로 값을 변경할 수 있다.

MaxJengineCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 32
- JSV 타입의 서버인 경우 8이 기본값이며 그 이외의 서버는 0이 기본값이 된다.
- 하나의 서버에 최대 몇 개의 Jengine이 연결을 맺는지에 대한 설정이다.

RequestLevelPing

- 종류: Boolean
- 기본값: N
- 서버 타입이 JSV인 경우 요청을 포워딩하기 전 해당 서버에 PING을 보내볼 것인지 설정한다.

- Y로 설정한 경우 JSV 서버로 포워딩하는 모든 요청에 대해 PING을 보내고 이에 대한 응답을 받은 경우에만 요청을 포워딩 한다.
- 여러 JSV 서버가 연결된 경우 OOM(Out Of Memory)등으로 요청을 보내도 응답하기 어려운 서버가 발생할 수 있는 경우에 사용하면 유용할 수 있다.

모든 요청에 대해 PING을 보내고 응답을 받아야 하므로 성능 저하가 발생할 수 있다. 기본값 사용을 권장한다.

RequestLevelPingTimeout

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 3
- 스케줄링된 JSV 서버로 PING을 보내고 이에 대한 응답을 받기까지 기다리는 시간을 설정한다. 해당 시간 안에 응답을 받지 못하면 해당 연결을 끊고 다른 JSV 서버를 다시 스케줄링을 하여 PING을 보내게 된다.

RequestLevelPingRetryCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- JSV 서버 내에 모든 Jengine에 대해 PING을 보내고 실패한 경우 재시도하고자 하는 횟수를 설정한다.

3.7.2. 예제

다음은 SERVER 절을 설정한 예제이다.

```
*SERVER
cgi
    SvgName = cgig,
    MinProc = 1,
    MaxProc = 5
ssi
    SvgName = ssig,
    MinProc = 1,
    MaxProc = 2
jsv1
    SvgName = jsvg,
    MinProc = 1,
    MAXProc = 10,
    LBFactor = 10
jsv2
    SvgName = jsvg,
    MinProc = 1,
    MAXProc = 10
```

```
    LBFactor = 5
jsv3
    SvgName = jsvg,
    MinProc = 1,
    MAXProc = 10
    LBFactor = 1
jsv4
    SvgName = jsvg,
    MinProc = 1,
    MAXProc = 10
wbaps
    SvgName = wbapg,
    MinProc = 2,
    MAXProc = 5,
    Schedule = "FA"
```

SvgName이 jsvg이고 *SVRGROUP 절에 LBServers로 설정된 jsv1, jsv2, jsv3가 로드 밸런싱 서버로 동작하고 각각 10, 5, 1의 비율로 요청을 분산한다(LBFactor는 LBType이 Static인 경우에만 의미가 있음). LBBackup인 jsv4는 jsv1, jsv2, jsv3가 모두 사용할 수 없을 경우에 요청을 처리한다.

3.8. SERVICE 절

SERVICE 절은 WebtoB를 통해 비즈니스 로직을 바로 수행할 경우 설정한다. 설정할 서비스의 서버 타입은 보통 WEBSTD로 설정한다. 기존의 표준 CGI를 구성하는 함수를 WebtoB에서 제공하는 함수로 사용해서 새로운 서비스 형태로 생성한다. 생성된 서비스는 WebtoB에서 바로 수행된다.

SERVICE 절에는 다음과 같은 내용이 정의된다.

- 서비스가 제공되는 서버 프로세스
- 서비스 우선순위
- 서비스 처리 제한시간

3.8.1. 설정 항목

다음은 SERVICE 절의 환경설정 형식이다.

```
*SERVICE
name    # String[16]
    SvrName = String[32],
   #Priority = Numeric,                # 50 (0-100)
   #SvcTime = Numeric                  # 0 (0-)
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

SERVICE 절 이름

- 종류: String
- 범위: 15자 이내
- WebtoB를 통해 수행할 비즈니스 로직에 해당하는 서버 프로그램 내의 함수 이름(서비스 루틴명)을 설정한다.
- 15자 이내의 string으로 반드시 SERVICE 절 내에서 유일(Unique)한 이름이어야 한다.

SvrName(필수 항목)

- 종류: String
- 범위: 31자 이내
- 해당 서비스를 제공하는 서버 프로세스를 지정한다. 해당 서비스 루틴을 가진 서버 프로그램의 실행 파일 이름을 설정한다.
- 서버 프로세스는 SERVER 절에 등록되어 있어야 한다.

Priority

- 종류: Numeric
- 범위: 0 ~ 100
- 기본값: 50
- 클라이언트의 요구를 처리하는 우선순위 값이다. 1부터 100까지 설정이 가능하며 숫자가 클수록 높은 우선순위를 갖는다.

SvcTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 무제한을 의미한다.)
- 서비스 처리의 제한시간이다. 각 서비스는 서비스 처리가 시작되는 순간부터 끝날 때까지 지정된 SvcTime 시간 안에 처리되어야 한다. 지정 시간을 초과하면 서버 프로세스는 서비스를 중지하고, 클라이언트에게 에러를 응답한다.

3.8.2. 예제

다음은 SERVICE 절을 설정한 예제이다.

```
*SERVICE
example
```

```
    SvrName = webaps,
write_board
    SvrName = webaps
```

3.9. DIRECTORY 절

노드 내의 특정 디렉터리의 속성을 설정한다. 디렉터리 접근에 인증이 필요하도록 하는 AuthentName, 디렉터리 안의 파일 확장명을 설정하는 ForceMimetype, DefaultMimetype 등이 있으며, 디렉터리 접근 내역을 기록하는 로그를 설정할 수 있다.

3.9.1. 설정 항목

다음은 DIRECTORY 절의 환경설정 형식이다.

```
*DIRECTORY
name    # String[32]
    Directory = Literal[256],           # $ENV, R.PATH
   #DefaultMimetype = Literal[128],
   #ForceMimetype = Literal[128],
   #VhostName = String[1024],           # LIST[64], MULTI
   #AccessName = String[32],
   #AuthentName = String[32],
   #Options = Literal[256],             # PM.LIST
   #DefaultCharset = Literal[32],
   #ErrorDocument = Literal[256]        # LIST[64]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

DIRECTORY 절 이름

- 종류: String
- 범위: 31자 이내
- 디렉터리의 이름을 설정한다.

Directory(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 설정을 적용할 디렉터리의 경로명을 설정한다. 경로명은 절대 경로와 $WEBTOBDIR을 기준으로 한 상대 경로를 사용할 수 있다.

DefaultMimetype

- 종류: Literal
- 범위: 127자 이내
- MIME-Type을 결정할 수 없는 문서의 Default Content-Type을 설정한다.
- 여러 절에서 적용되는 우선순위는 다음과 같다.
  1. DIRECTORY 절에 설정된 DefaultMimetype
  2. SVRGROUP 절에 설정된 DefaultMimetype
  3. 해당되는 NODE 절 또는 VHOST 절에 설정된 DefaultMimetype

ForceMimetype

- 종류: Literal
- 범위: 127자 이내
- 지정된 디렉터리 안의 모든 리소스들은 ForceMimetype에 설정한 MIME-Type으로 처리된다.

  예를 들어 ForceMimetype이 CGI로 되어있다면 디렉터리 내의 모든 리소스들은 클라이언트의 요구가 있을 경우 CGI로 처리된다. 즉, CGI 처리를 담당하고 있는 서버가 서비스를 처리한다.

VhostName

- 종류: Literal
- 범위: 1023자 이내
- Virtual Host를 이용하게 구성된 환경에서 디렉터리에 대한 설정을 해당 Virtual Host에만 적용되도록 해야 할 필요가 있을 때 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.
- VhostName 항목을 설정하지 않는 경우 모든 Virtual Host에 적용 가능하다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP 주소에서 들어온 요청을 허용 또는 거부하는 ACCESS 절의 설정을 DIRECTORY 절에 할 수 있다.

AuthentName

- 종류: String

- 범위: 31자 이내
- DIRECTORY 절에 대해서 인증(Authentication)을 적용할 수 있다. 적용할 DIRECTORY 절에 AUTHENT 절에서 설정한 이름을 설정한다. 단, 서버 타입이 JSV인 경우 SVRGROUP 절에만 인증을 적용할 수 있다.

Options

- 종류: Literal
- 범위: 255자 이내
- 다음은 설정 가능한 옵션에 대한 설명이다.

옵션	내용
{+|-}Cache	Content의 캐시 유무(+|-)를 결정한다.
SSLRequireSSL	SSL을 사용하는 요청만 처리하고, SSL을 사용하지않는 요청은 "403 Forbidden"으로 응답한다.
SSLDenySSL	SSL을 사용하지 않는 요청만 처리하고, SSL을 통한 요청은 "403 Forbidden"으로 응답한다.

DefaultCharset

- 종류: Literal
- 범위: 31자 이내
- HTTP Header 중에서 Content-Type에 Character set 관련 파라미터가 없는 응답에 추가될 Character set의 이름을 설정한다.

설정 값	내 용
On	기본 Character set인 ISO-8859-1로 설정한다.
Off	해당 기능을 중단한다.
charset	사용자가 기술한 '_charset_'으로 설정한다.

- 여러 절에서 적용되는 우선순위는 다음과 같다.
  1. DIRECTORY 절에 설정된 DefaultCharset
  2. SVRGROUP 절에 설정된 DefaultCharset
  3. 해당되는 NODE 절 또는 VHOST 절에 설정된 DefaultCharset

ErrorDocument

- 종류: Literal
- 범위: 255자 이내

- HTTP 에러 페이지를 사용자가 지정한 페이지로 대신 사용할 경우 ERRORDOCUMENT 절에 정의한 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 설정할 수 있다.

3.9.2. 예제

다음은 DIRECTORY 절을 설정한 예제이다.

```
*DIRECTORY
dir_test
    DIRECTORY = "/usr/local/webtob/docs/vhost_docs",
    ForceMimetype = "text/plain"
```

3.10. URI 절

URI 절은 클라이언트 요구의 URI(Uniform Resource Identifier) 값에 따라 이를 처리하는 서비스를 구분할 수 있도록 한다. 특정 URI가 입력으로 들어온 경우 이를 특정 서비스에서 처리하도록 할 수 있다. 보통 URI 절은 CGI를 이용하는 경우에 많이 사용된다.

예를 들어 사용자가 http://www.tmax.co.kr/cgi-bin/test.cgi를 호출한 경우 "/cgi-bin/" URI를 CGI의 서비스로 정의해서 사용할 수 있다.

단, 아래와 같은 요청이 있을 때 URI 절 설정은 <first> 또는 <first>로 시작하는 URI로 해야 한다.

```
http://<hostname>:<port>/<first>/<second>/index.html
```

3.10.1. 설정 항목

다음은 URI 절의 환경설정 형식이다.

```
*URI
name    # String[32]
    URI = Literal[256],
    SvrType = String[32],
   #SvrName = String[32],
   #SvcName = String[16],
   #Redirect = Literal[256],
   #RedirectStatus = String[32],        # 302
   #VhostName = String[1024],           # LIST[64], MULTI
   #AccessName = String[32],
   #AuthentName = String[32],
   #Options = Literal[256],             # PM.LIST
   #SCGI = Boolean,                     # N
   #SCGIServer = Literal[256],
   #Match = Literal[256],               # "prefix"
   #Priority = Numeric,                 # 50 (0-100)
   #FCGI = Boolean,                     # N
   #RedirectNoSub = Boolean,            # N
```

```
#Ext = String[256],                 # LIST, MULTI
#GotoEXT = Boolean,                 # N
#StopIfNoEXT = Boolean,             # Y
#LBSvgName = Literal[256]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

URI 절 이름

- 종류: String
- 범위: 31자 이내
- string으로 사용자 임의로 정할 수 있다.

Uri(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- HTTP Request path와 매치할 패턴을 설정한다. 매치되면 해당 요청은 URI 절의 설정이 적용된다.
- WebtoB는 여러 가지 패턴 종류를 지원한다. 패턴 종류는 Match 항목에 설정한다.

SvrType(필수 항목)

- 종류: String
- 범위: 31자 이내
- 서비스 속성 즉, 지정된 URI를 포함하는 요청이 왔을 때 처리할 서버를 설정한다. 예를 들어 "/jsv/"라는 URI를 포함하는 요청에 대해 서버 타입이 JSV인 서버가 지정되도록 한다.

SvrName

- 종류: String
- 범위: 31자 이내
- SvrName은 처리 담당 서버의 이름을 설정한다.
- WebtoB에서는 같은 서버 타입을 갖고 있는 서비스 개체들이 서버 그룹과 서버로 구분될 수 있고, 각각의 서버는 서비스 처리 프로세스의 최소 및 최대 개수를 설정한다.
- URI 절에서는 처리를 담당할 서버를 지정하여 세분화된 서비스 제어를 할 수 있다.

SvcName

- 종류: String
- 범위: 15자 이내
- 해당 URI에 대한 처리를 전담할 서비스의 이름을 설정한다.

Redirect

- 종류: String
- 범위: 255자 이내
- 지정 URI에 대한 요구를 다른 URI에 매핑시키도록 하는 기능이다. URI 절에 설정된 RedirectionStatus의 값에 따라 Redirect에 설정된 값이 HTTP 응답의 Location Header 필드에 설정되어 사용자에게 전달된다.
- RedirectStatus의 값을 생략하고 Redirect만 사용된 경우 그 값은 "302 Found"를 사용한다.

RedirectStatus

- 종류: String
- 범위: 31자 이내
- 기본값: 302
- Redirect 기능을 사용하는 경우 발생될 HTTP status를 설정한다.
- 다음과 같은 값을 사용할 수 있다.

설정값	별칭	설명
301	permanent	"301 Moved Permanently"로 응답한다.
302	found	"302 Found"로 응답한다.
303	seeother	"303 See Other"로 응답한다.
305	useproxy	"305 Use Proxy"로 응답한다.
307	temp	"307 Temporary Redirect"로 응답한다.
410	gone	"410 Gone"로 응답한다.

VhostName

- 종류: Literal
- 범위: 1023자 이내
- URI가 특정 Virtual Host에 사용될 때 VHOST 절에 정의된 해당 Virtual Host 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP 주소에서 들어온 요청을 허용 또는 거부하는 ACCESS 절 설정을 URI 절에 적용할 수 있다.

AuthentName

- 종류: String
- 범위: 31자 이내
- URI 절에 대해서 인증(Authentication)을 적용할 수 있다. 적용할 URI에 AUTHENT 절에서 설정한 이름을 설정한다. 단, 서버 타입이 JSV인 경우 SVRGROUP에만 인증을 적용할 수 있다.

Options

- 종류: Literal
- 범위: 255자 이내
- 다음은 적용 가능한 옵션에 대한 설명이다.

옵션	내용
{+|-}Cache	Content의 캐시 유무(+|-)를 결정한다.
SSLRequireSSL	SSL을 사용하는 요청만 처리하고, SSL을 사용하지않는 요청은 "403 Forbidden"으로 응답한다.
SSLDenySSL	SSL을사용하지않는 요청만 처리하고, SSL을 통한 요청은 "403 Forbidden"으로 응답한다.

SCGI

- 종류: Boolean
- 기본값: N
- 'SvrType = CGI'인 경우 요청이 Simple CGI인지 설정한다.
- Y로 설정하면 요청을 Simple CGI 방식으로 처리한다. 이 경우 SCGIServer도 함께 설정해야 한다.

SCGIServer

- 종류: Literal

- 범위: 255자 이내
- Simple CGI의 서버 주소를 설정한다.
- 형식은 "서버이름(IP 주소):포트 번호"이다.

```
SCGIServer = "172.16.1.100:9001"
```

Match

- 종류: Literal
- 범위: 255자 이내
- 기본값: "prefix"
- URI에 설정된 패턴의 종류를 설정한다.

설정값	설명
prefix	URI에 설정된 패턴이 HTTP Request URL의 prefix이다. (예: 패턴 "/uri/"는 "/uri/a/", "/uri/a/b/", "/uri/a/b/c" 등의 Request path와 매치한다.)
exact	URI에 설정된 패턴이 HTTP Request URL과 일치한다. (예: 패턴 "/uri/"는 Request path가 "/uri/"일 경우만 매치한다. 이외 모든 Request path는 매치하지 않는다.)
fn	URI에 설정된 패턴이 UNIX 셸에서 사용되는 FNMATCH 패턴이다. 애스터리스크(*), 물음표(?) 같은 와일드카드를 사용할 수 있다. (예: 패턴 "/uri/*/*/"는 "/uri/a/b/", "/uri/c/d/" 등의 Request path와 매치한다.)
regexp	URI에 설정된 패턴이 Perl 스크립트 언어에서 사용되는 regular expression이다. (예: 패턴 "/uri.*"은 "/uri", "/uri/", "/uri1"와 같은 Request path와 매치한다.)

- 패턴 종류에 따라서 HTTP Request path와 매치하는 방식이 달라진다.

Priority

- 종류: Numeric
- 범위: 0 ~ 100
- 기본값: 50
- 클라이언트의 요구를 처리하는 우선순위 값이다. 1부터 100까지 설정이 가능하며 숫자가 클수록 높은 우선순위를 갖는다.

FCGI

- 종류: Boolean
- 기본값: N
- 'SvrType = CGI'인 경우 요청이 Fast CGI인지 설정한다.
- Y로 설정하면 요청을 Fast CGI 방식으로 처리한다.

RedirectNoSub

- 종류: Boolean
- 기본값: N
- Redirect 설정을 하는 경우 prefix로 매칭할 경우 설정된 URL에 요청 URL의 나머지 부분을 붙여서 redirect할지 여부를 설정한다.

Ext

- 종류: Literal
- 범위: 255자 이내
- URI 매칭하는 경우 추가적으로 설정한 확장자까지 매칭한다.
- 특정 경로 아래의 특정 확장자에 대해서만 URI 절을 적용할 경우 사용한다.
- 콤마(,)를 사용하거나 여러 번 설정하여 여러 확장자를 설정할 수 있다.

GotoExt

- 종류: Boolean
- 기본값: N
- URI 매칭을 먼저하는 경우 EXT 절 매칭을 수행하도록 할지 여부를 설정한다. EXT 매칭이 되면 해당 EXT 서비스가 선택되는 것이므로 앞서 매칭한 URI 설정은 적용되지 않음을 주의해야 한다.

StopIfNoExt

- 종류: Boolean
- 기본값: Y
- 'GotoExt = Y'인 경우 적용된다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	매칭되는 EXT 절 설정 항목이 없을 경우 매칭을 중단한다.
N	다음 URI 절 설정을 계속 탐색한다.

LBSvgName

- 종류: Literal
- 범위: 255자 이내
- 지정된 URI를 포함하는 요청이 왔을 때 이를 처리할 서버들 사이에 부하 분산을 설정하고자 할 경우 원하는 SVRGROUP명을 설정한다.
- LBSvgName에 명시한 SVRGROUP 설정에는 LBServers나 LBBackup 옵션이 적용되어야 한다.

3.10.2. 예제

다음은 URI 절을 설정한 예제이다.

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

3.11. ALIAS 절

실제 서버의 물리적 디렉터리 경로와 URI를 Alias시키도록 설정할 수 있다. 즉, 어떤 특정한 URI에 대한 요구가 들어오면 이를 실제의 물리적인 디렉터리에 매핑시켜서 이곳에서 원하는 리소스를 찾아 처리하게 하는 방식이다. 이는 사용자가 Document root에 상관없이 지정할 수 있기 때문에 관리하는 입장에서 편리한 기능이다.

3.11.1. 설정 항목

다음은 ALIAS 절의 환경설정 형식이다.

```
*ALIAS
name    # String[32]
    URI = Literal[256],
    RealPath = Literal[256],            # $ENV, R.PATH
```

```
#VhostName = String[1024]           # LIST[64], MULTI
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

ALIAS 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

URI(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- Alias할 URI를 설정한다.

Realpath(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 서버 안의 물리적 디렉터리의 경로명을 설정한다.
- 대량의 Virtual Host를 사용하는 경우 다음의 지시자를 사용해서 디렉터리 설정 패턴을 동적으로 설정할 수 있다.

지시자	설명
%p	요청한 포트 번호로 치환한다.
%n	Hostname이나 IP 주소의 n번째 요소로 치환한다. 만약 n을 0으로 하면 전체 문자열이 사용된다. 마이너스 기호(-)가 앞에 오면 Hostname이나 IP 주소의 끝에서부터 센다. 플러스 기호(+)가 뒤에 오면 Hostname이나 IP 주소의 나머지가 사용된다.
%n.m	n번째 요소의 m번째 문자로 치환한다. 위와 같이 마이너스 기호(-)나 플러스 기호(+)가 붙을 수 있다.
%%	단일 퍼센트(%) 표시로 치환한다.

VhostName

- 종류: Literal

- 범위: 1023자 이내
- ALIAS 절이 적용되는 Virtual Host를 제한하고 싶은 경우 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

3.11.2. 예제

다음은 ALIAS 절을 설정한 예제이다.

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

3.12. DIRINDEX 절

요청 주소가 디렉터리까지만 명시된 경우 기본적으로 해당 디렉터리에서 index.html을 찾아 보여주도록 되어 있다. 그러나 index.html이 없는 경우 해당 디렉터리 구조를 보여줄 수 있도록 설정할 수 있다. 또한 그것을 인덱싱하는 방식과 Icon 등을 지정할 수 있다.

3.12.1. 설정 항목

다음은 DIRINDEX 절의 환경설정 형식이다.

```
*DIRINDEX
name    # String[32]
    Options = Literal[256],             # "Fancy,EncodeURL", PM.LIST
   #Ignore = Literal[256],              # LIST
   #DefaultIcon = String[32],
   #Description = Literal[256],
   #HeaderFile = String[32],
   #TailFile = String[32],
   #IconExt = Literal[256]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

디렉터리 인덱싱 기능을 사용하기 위해서 DIRINDEX 절에 다음 항목 설정이 필요하다.

항목	설명
Options	“+Index”로 설정하면 디렉터리 정보를 보여준다.
DirIndex	DIRINDEX 이름을 적어준다.

DIRINDEX 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

Options(필수 항목)

- 종류: Literal
- 인덱싱하는 방식에 대한 옵션을 사용하여 설정할 수 있다.
- 다음은 설정 옵션에 대한 설명이다.

옵션	설명
{+|-}Fancy	WebtoB가 제공하는 Fancy Indexing 사용을 결정한다. Fancy Indexing은 file name, last modified time, file size 별로 정렬하는 링크를 보여주는 기능이다.
{+|-}EncodeURL	생성된 페이지 내부의 URL 중 ASCII가 아닌 글자들에 대한 URL 인코딩 사용을 결정한다. "-EncodeURL"로 설정할 경우 한글 파일의 링크를 한글 그대로 보이게 되며, 브라우저 설정에 따라 해당 파일이 Not Found가 될 수 있기 때문에 주의해야 한다.

Igore

- 종류: Literal
- 범위: 255자 이내
- 인덱싱할 때 필요에 따라 표시하지 않을 파일들이 Index 리스트에 나타나지 않도록 해준다. 인덱싱에서 제외하고 싶은 파일의 리스트를 파일명을 써주거나 애스터리스크(*)를 사용해서 설정한다.

DefaultIcon

- 종류: String
- 범위: 31자 이내
- Unknown File Type에 대한 Icon을 설정한다.

Description

- 종류: Literal
- 범위: 255자 이내

- 디렉터리 내 파일들에 대한 Description을 명시한 파일명을 설정한다. 이 설정은 디렉터리 구조를 표시할 때 각 파일들의 Description을 보여주도록 한다.

  예를 들어 해당 디럭터리에 a.html, b.html, c.html, .. 이 존재할 때 각 파일에 대한 설명을 출력하려면 des.txt 파일 내용을 다음과 같이 작성하고 Descrition = "des.txt"로 설정한다.

  ```
  a.html a.html에 대한 설명
  b.html b.html에 대한 설명
  c.html c.html에 대한 설명
  ...
  ```

HeaderFile

- 종류: String
- 범위: 31자 이내
- 인덱싱의 가장 윗부분에 집어 넣을 Header를 지정된 파일에서 읽어 올 수 있도록 한다. 파일명은 인덱싱하고 있는 디렉터리에서 상대적인 경로명으로 간주한다.

TailFile

- 종류: String
- 범위: 31자 이내
- 인덱싱의 가장 아래 부분에 집어 넣는 내용을 TailFile에서 지정한 파일에서 읽어 온다. 파일명은 인덱싱하고 있는 디렉터리에서 상대적인 경로명으로 간주한다.

IconExt

- 종류: Literal
- 범위: 255자 이내
- 파일의 Extension 값에 따라 해당 Icon으로 인덱싱을 할 수 있도록 설정한다.
- Icon 파일의 URL과 MIME-Type을 매칭시킨다.

3.12.2. 예제

다음은 DIRINDEX 절을 설정한 예제이다.

```
*DIRINDEX
dindex
    Options = "Fancy"
```

3.13. LOGGING 절

클라이언트의 요구 내역을 기록하는 형식을 지정한다. 접근 내역과 에러 내역이 따로 저장되며 저장 형식을 지정할 수 있다. 시스템 로그, 액세스 로그, 에러 로그 모두 LOGGING 절에 설정한다.

3.13.1. 설정 항목

다음은 LOGGING 절의 환경설정 형식이다.

```
*LOGGING
name    # String[32]
    FileName = Literal[256],            # $ENV, R.PATH
    Format = Literal[256],
   #Option = Literal[256],               # PM.LIST
   #RotateBySeconds = Numeric,          # 0 (0-)
   #ExcludeByExt = Literal[1024],       # LIST
   #ArchiveFileName = Literal[256],     # $ENV
   #ValidHours = Numeric,               # 0 (0-23)
   #LogHandler = Literali[256]          # LIST[5]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

다음은 LOGGING 절의 설정과 관계되는 NODE 절의 항목이다. SysLog를 제외한 Logging, Errorlog는 NODE 절 외 VHOST 절에서도 설정할 수 있다.

항목	설명
SysLog	시스템 로그 이름을 설정한다.
Logging	액세스 로그 이름을 설정한다.
Errorlog	에러 로그 이름을 설정한다.

LOGGING 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

FileName(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 로그를 저장할 파일의 경로를 설정한다.

- 상대 경로 ("/"로 시작하지 않는 경로)는 "$WEBTOBDIR/상대 경로"로 자동 대체한다.
- 파일 이름에 포함된 다음 substitution string은 파일이 생성될 때 실제 값으로 대체된다. FileName에 설정된 값은 "$WEBTOBDIR/log/access_20091105.log"로 변경된다.

```
FileName = "log/access_%Y%%M%%D%.log"
```

다음은 substitution string의 실제값에 대한 설명이다.

String	실제값
%Y%	년도(year)로 숫자 4개로 설정한다. (예: 2009)
%M%	월(month)로 숫자 2개를 설정한다. (예: 11)
%D%	일(day)로 숫자 2개를 설정한다. (예: 05)
%h%	시(hour)로 숫자 2개를 설정한다. (예: 10)
%m%	분(minute)으로 숫자 2개를 설정한다. (예: 30)
%s%	초(second)로 숫자 2개를 설정한다. (예: 45)

Format(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 로그 파일에 기록될 메시지의 포맷을 설정한다.
- *USERLOGFORMAT 절에서 정의한 로그 포맷 이름으로 설정 할 수 있다.
- 다음 Format string은 액세스 로그, 에러 로그 및 Header 절의 FieldValue에만 적용된다. 시스템 로그는 임의 값을 설정한다.

포맷	설명
DEFAULT	Default Log File Format이다. (약자: "%h %t \"%r\" %s %b %D")
COMMON	Common Log File Format이다. (약자: "%h %l %u %t \"%r\" %s %b")
COMBINED	Combined Log File Format이다. (약자: "%h %l %u %t \"%r\" %s %b \"%{*Referer*}i\" \"%{*User-Agent*}i\"")
COMBINEDIO	CombinedIO Log File Format이다. (약자: "%h %l %u %t \"%r\" %s %b \"%{*Referer*}i\" \"%{*User-Agent*}i\" %I %O ")
%a	요청을 보낸 장비의 IP 주소를 표시한다. %h와 동일하다.

포맷	설명
%b	헤더를 제외한 응답의 Byte를 표시한다.
%c	WebtoB 응답 생성 위치를 표시한다. • 내부 캐시에서 응답이 생성된 경우 "hc" 로 표시한다. • 디스크 캐시에서 응답이 생성된 경우 "dc"로 표시한다. • sendfile에서 응답이 생성된 경우 "sf"로 표시한다. • sendfile / disk cache 에서 생성된 경우 "sf/dc"로 표시한다. • remote 에서 생성된 경우 "hm"로 표시한다.
%{_attr_name_}C	HTTP Request의 Cookie Header 값 중 '_attr_name_'에 해당하는 값을 표시한다.
%d	응답이 전송된 시간을 표시한다.
%D	요청을 처리하는데 소요된 시간을 표시한다. (단위: millisecond)
%{ENV_NAME}e	환경변수 ENV_NAME을 출력한다.
%g	WebtoB가 내부적으로 사용하는 요청 식별자를 출력한다.
%h	요청을 보낸 장비의 IP 주소를 표시한다.
%H	사용한 HTTP 버전을 표시한다.
%{HEADER_FIELD}i	HTTP Request의 HEADER_FIELD Header 값을 표시한다.
%{_id_name_}J	요청을 JEUS로 포워딩하여 처리하는 경우 내부적으로 사용하는 요청의 식별정보를 표시한다. • _id_name이 JSVCid이면 Client ID이다. • _id_name이 JSVReqSeq이면 Request Sequence이다.
%l	원격 로그인명을 표시한다.
%m	HTTP Request 메소드를 표시한다.
%p	Request가 도착한 서버의 포트 번호를 표시한다.
%q	HTTP Request의 query 값을 표시한다.
%r	HTTP Request의 Request line 전체를 표시한다.
%R	HTTP Request의 Request line 전체를 표시한다. CheckURL이나 URLRewrite 기능에 의해 변경된 Request line을 표시한다.
%s	응답에 사용된 HTTP Status Code를 표시한다.
%t	요청처리를 마친 시간을 표시한다.
%T	Request를 처리하는 데 소요된 시간을 표시한다. (단위: 초)
%u	HTTP 인증에 사용된 user 이름을 표시한다.
%U	HTTP Request URI를 표시한다.
%v	Host Header 필드 값을 표시한다.
%z	응답이 압축된 경우 압축 전/후의 응답 크기와 압축률을 표시한다.

포맷	설명
%S	http와 https를 구분하여 표시한다.
%A	서버의 IP 주소를 표시한다.
%I	요청의 byte를 표시한다.
%O	응답의 byte를 표시한다.

Option

- 종류: Literal
- 범위: 255자 이내
- Logging 방식에 영향을 주는 옵션들을 설정한다.

옵션	설명
Sync	로그 메시지들이 WebtoB 메모리에 버퍼링되지 않고 바로 파일에 기록되도록 한다. 증권업무나 은행업무 등 사용자의 로그를 바로 확인하여야 하는 곳에서는 이러한 옵션을 주어서 사용하면 문제가 발행하는 경우 쉽게 확인할 수 있는 장점이 있다.
NoErrorClientAddress	WebtoB가 로깅하는 에러 메세지에 client의 IP 주소가 포함된 경우 이를 제외한다. 사용자의 IP 정보를 노출하지 않음으로서 보안성을 높일 수 있다는 장점이 있다.

RotateBySeconds

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 이 기능을 사용하지 않는것을 의미한다.)
- 새로운 로그 메시지 생성할 때 현재 존재하는 로그 파일이 지정된 시간보다 오래되면 새로운 로그 파일을 생성하도록 설정한다.

ExcludeByExt

- 종류: Literal
- 범위: 1023자 이내
- 특정 확장자를 가진 요청 파일에 대해서 액세스 로그를 남기지 않는 경우 해당 확장자 이름을 설정한다.
- 다음은 설정에 대한 예이다.

```
ExcludeByExt = "jpg,png"
```

ArchiveFileName

- 종류: Literal
- 범위: 255자 이내
- ArchiveFileName을 설정하면 새로운 로그 파일 생성시점에 현재 로그 파일이 설정된 이름으로 이동한다. 그리고 새로운 로그 파일이 FileName에 설정한 형식(파일명+날짜)으로 생성된다. FileName과 같은 형식의 로그 파일 이름을 설정하면 된다.

ValidHours

- 종류: Numeric
- 단위: hours
- 범위: 0 ~ 23
- 기본값: 0 (0은 이 기능을 사용하지 않는것을 의미한다.)
- 새로운 로그 파일을 설정된 시간 단위로 생성하도록 설정한다.

LogHandler

- 종류: Literal
- 범위: 255자 이내
- access log를 리모드 서버에 남길 경우 사용할 LOG_HANDLER 절 이름을 설정한다.
- 콤마(,)로 구분하여 최대 5개까지 설정 가능하다.

3.13.2. 예제

다음은 LOGGING 절을 설정한 예제이다.

```
*LOGGING
access_log
    Format = "default", # "%h %l %u %t \"%r\" %s %b" is default format
    Filename = "log/archives/access_%Y%%M%%D%.log"
    Option = "sync"
error_log
    Format = "%r Host=%{HOST}i",
    Filename = "log/archives/error_%Y%%M%%D%.log"
    Option = "sync"
system_log
    Format = "",
    Filename = "log/archives/system_%Y%%M%%D%.log"
```

```
    Option = "sync"
```

3.14. ACCESS 절

클라이언트에서 접속을 시도할 때 IP 주소, network/netmask, Header 정보들을 기준으로 요청의 허용/제한을 설정한다. 또한 요청 허용/제한이 적용되는 순서를 설정할 수 있다.

ACCESS 절은 DIRECTORY 절, URI 절, EXT 절, TCPGW 절에 적용될 수 있으며, 각각에서 정의한 리소스를 허용/제한하게 된다.

3.14.1. 설정 항목

다음은 ACCESS 절의 환경설정 형식이다.

```
*ACCESS
name    # String[32]
   #Method = Literal[256],              # "<all-methods>", PM.LIST
   #MethodExcept = Literal[256],        # PM.LIST
   #Order = Literal[256],               # "deny,allow"
   #Allow = Literal[256*35],            # LIST[256]
   #Deny = Literal[256*35],             # LIST[256]
   #AllowIf = Literal[256],             # MULTI[32]
   #DenyIf = Literal[256]               # MULTI[32]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

ACCESS 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

Method

- 종류: Literal
- 범위: 255자 이내
- 기본값: <all-HTTP-methods>
- 적용할 HTTP 메소드를 설정한다.

MethodExcept

- 종류: Literal
- 범위: 255자 이내
- WebtoB에서 설정할 수 있는 HTTP 메소드(GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK)에서 제외하고 싶은 메소드들을 설정한다. 즉, 설정된 메소드들을 제외한 메소드만 적용된다.

  예를 들면 MethodExcept = "POST"는 Method = "GET, HEAD, ..."(POST 제외한 모든 HTTP 메소드)와 동일하다.

Order

- 종류: Literal
- 기본값: "Deny,Allow"
- Allow, AllowIf, Deny, DenyIf, Method가 적용되는 순서를 설정한다.
- 다음은 옵션에 대한 설명이다.

옵션	설명
Deny, Allow	Method > Deny > DenyIf > Allow > AllowIf 순서로 매치한다. • 요청 메소드가 Method 항목에 포함되지 않을 경우: 요청을 허용한다. • Deny, DenyIf와 매치하지만 Allow, AllowIf와 매치하지 않은 경우: 요청을 거절한다. • 이외 모든 경우: 요청을 허용한다.
Allow, Deny	Method > Allow > AllowIf > Deny > DenyIf 순서로 매치한다. • 요청 메소드가 Method 항목에 포함되지 않을 경우: 요청을 거절한다. • Allow, AllowIf와 매치하지만 Deny, DenyIf와 매치하지 않은 경우: 요청을 허용한다. • 이외 모든 경우: 요청을 거절한다.

Allow

- 종류: Literal
- 범위: 8959자 이내
- 요청이 허용되는 IP 주소나 Network/Netmask들을 설정한다.
- IP 주소와 Network/Netmask를 다음과 같이 설정한다.

```
Allow = "192.168.1.43/255.255.255.0"
```

- Allow = "all"은 특수 값으로서 모든 IP 주소를 의미한다.

- 콤마(,)로 구분하여 최대 256개까지 설정할 수 있다.

Deny

- 종류: Literal
- 요청이 거절되는 IP 주소나 Network/Netmask들을 설정한다. 설정 방식은 Allow와 동일하다.

AllowIf

- 종류: Literal
- 범위: 255자 이내
- “<header field name> <regular expression>”으로 설정한다.
- 요청에 포함된 <header field name> Header 값이 <regular expression>의 패턴과 매치된다. <regular expression>은 Perl compatible regular expression을 사용한다.
- 1개 이상의 AllowIf 설정도 가능하다.

```
AllowIf="Referer http://10.0.0.2/"
```

DenyIf

- 종류: Literal
- 범위: 255자 이내
- AllowIf와 동일한 방식으로 설정한다.

3.14.2. 예제

다음은 ACCESS 절을 설정한 예제이다.

```
*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 = "deny, allow",
```

```
    Deny = "211.1.1.30"

access5
    Order = "allow, deny",
    Allow = "all",Deny = "211.1.1.30"
```

3.15. AUTHENT 절

클라이언트의 접근을 제한하기 위한 인증 과정을 사용자와 그룹 단위로 통제할 수 있도록 설정한다.

AUTHENT 절은 SVRGROUP 절, URI 절, EXT 절, DIRECTORY 절에 설정되어 인증 과정을 거칠 수 있다.

3.15.1. 설정 항목

다음은 AUTHENT 절의 환경설정 형식이다.

```
*AUTHENT
name    # String[32]
    Type = String[32],
    UserFile = Literal[256],            # $ENV, R.PATH
    #AccessName = String[32]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

AUTHENT 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

Type(필수 항목)

- 종류: String
- 범위: 31자 이내
- Authorization control(인가 제어) 방식을 설정한다. Basic과 Digest 방식을 지원한다. 자세한 설정은 인증을 참고한다.

UserFile(필수 항목)

- 종류: Literal

- 범위: 255자 이내
- 인증을 실행하기 위한 사용자명과 암호명이 기록되는 파일을 지정한다.
- WebtoB는 UserFile 관리의 편의를 돕기 위해 **'wsmkpw'**라는 유틸리티를 제공한다. wsmkpw를 사용하여 사용자명과 암호화(encrypted)된 암호명을 UserFile에 기록할 수 있다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP 대역별로 들어온 요청에 대해 인가 제어를 하는 경우 적용 할 ACCESS 절 이름을 설정한다.

3.15.2. 예제

다음은 AUTHENT 절을 설정한 예제이다.

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

3.16. EXT 절

클라이언트가 요구한 파일의 확장자명에 따라 처리 담당 프로세스를 설정한다. WebtoB는 기본적인 모든 MIME-Type에 대한 처리 담당 프로세스가 설정되어 있으나, 필요에 따른 추가적인 설정을 할 경우 이 절에서 할 수 있다.

3.16.1. 설정 항목

다음은 EXT 절의 환경설정 형식이다.

```
*EXT
name    # String[32]
   #Mimetype = Literal[128],
   #SvrType = String[32],               # HTML
   #SvrName = String[32],
   #SvcName = String[16],
   #VhostName = String[1024],           # LIST[64], MULTI
   #AccessName = String[32],
   #AuthentName = String[32],
   #Options = Literal[256],             # PM.LIST
   #Charset = Literal[32],
   #SCGI = Boolean,                     # N
   #ScgiServer = Literal[256],
   #Match = Literal[256],               # "exact"
   #RegExp = Literal[512],
   #Priority = Numeric,                 # 50, 0 ~ 100
   #FCGI = Boolean,                     # N
```

```
#Extension = Literal[32]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

EXT 절 이름

- 종류: String
- 범위: 31자 이내
- HTTP Request path의 확장자와 매치할 패턴을 설정한다. 매치되면 해당 요청은 EXT 절의 설정이 적용된다.
- WebtoB는 여러 가지 패턴 종류를 지원한다. 패턴 종류는 Match 항목에 설정한다.

Mimetype

- 종류: Literal
- 범위: 127자 이내
- 확장자에 해당하는 MIME-Type을 설정한다.

```
MimeType = "text/html"
```

SvrType

- 종류: String
- 기본값: HTML
- 확장자를 처리할 서버의 타입을 설정한다.

SvrName

- 종류: String
- 범위: 31자 이내
- 특정한 서버가 처리하도록 설정할 경우 서버의 이름을 설정한다.

SvcName

- 종류: String
- 범위: 15자 이내

- 특정 서비스에서 처리하도록 설정할 경우 해당 서비스의 이름을 설정한다.

VhostName

- 종류: String
- 범위: 1023자 이내
- EXT가 특정 Virtual Host에 사용될 때 VHOST 절에 정의된 해당 Virtual Host 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP에서 들어온 요청을 허용 또는 거부하는 설정을 EXT 절에 설정할 수 있다.

AuthentName

- 종류: String
- 범위: 31자 이내
- EXT 절에 대해서 인증(Authentication)을 적용할 수 있다. 적용할 EXT에 AUTHENT 절에서 설정한 이름을 설정한다. 단, 서버 타입이 JSV인 경우 SVRGROUP에만 인증을 적용할 수 있다.

Options

- 종류: Literal
- 다음은 설정 가능한 옵션에 대한 설명이다.

옵션	설명
{+|-}Cache	Content의 캐시 유무(+|-)를 결정한다.
SSLRequireSSL	SSL을 통한 Request만 처리하고, Non-SSL을 통한 Request는 "403 Forbidden"으로 응답한다.
SSLDenySSL	Non-SSL을 통한 Request만 처리하고, SSL을 통한 Request는 "403 Forbidden"으로 응답한다.
UnSet	기본으로 설정되어 있는 CGI, JSP 등에 대해서 해당 확장자를 EXT 절에서 Unset시킨다. html의 경우 웹 서버의 가장 기본적인 기능이기 때문에 Unset시킬 수 없다.

Charset

- 종류: Literal
- Content-Type Header에 추가할 Charset을 설정한다.

설정 값	설명
on	"iso-8859-1"이 사용된다.
off	Content-Type Header에 Charset을 추가하지 않는다.
<*custom*>	사용자가 설정한 값을 Charset으로 사용한다.

SCGI

- 종류: Boolean
- 기본값: N
- 'SvrType = CGI'인 경우 요청이 Simple CGI인지 여부를 설정한다.
- Y로 설정하면 요청을 Simple CGI 방식으로 처리한다. 이때, SCGIServer도 함께 설정해야 한다.

ScgiServer

- 종류: Literal
- 범위: 255자 이내
- Simple CGI 서버 주소를 설정한다.
- 형식은 "서버 이름(IP 주소):포트 번호"이다.

```
SCGIServer = "172.16.1.100:9001"
```

Match

- 종류: Literal
- 기본값: "exact"
- EXT name에 설정된 패턴의 종류를 설정한다. 패턴 종류에 따라서 HTTP Request path의 확장자와 매치하는 방식이 달라진다.
- 다음은 설정 가능한 패턴의 종류에 대한 설명이다.

패턴 종류	설명
prefix	설정된 패턴이 확장자의 prefix이다. (예: 패턴 "abc"는 "abc", "abc1", "abc12" 등의 확장자와 매치한다.)

패턴 종류	설명
exact	설정된 패턴이 확장자와 일치한다. (예: 패턴 "abc"는 확장자가 "abc"일 경우만 매치한다. 이외 모든 확장자는 매치하지 않는다.)
regexp	설정된 패턴이 Perl 스크립트 언어에서 사용되는 Regular expression 이다. regexp를 사용하려면 패턴을 RegExp에 설정해야 한다. 이 경우 EXT name의 패턴은 무시된다. (예: 패턴 "jsp[a-z]"은 "jspa", "jspb", "jspx"와 같은 확장자와 매치한다.)

RegExp

- 종류: Literal
- 범위: 511자 이내
- Regular expression을 사용한 패턴을 설정한다.
- Match = "regexp"인 경우만 적용된다.

Priority

- 종류: Numeric
- 범위: 0 ~ 100
- 기본값: 50
- 클라이언트의 요구를 처리하는 우선순위 값을 설정한다. 1부터 100까지 설정이 가능하며 숫자가 클수록 높은 우선순위를 갖는다.

FCGI

- 종류: Boolean
- 기본값: N
- 'SvrType = CGI'인 경우 요청이 Fast CGI인지 설정한다.
- Y로 설정하면 요청을 Fast CGI 방식으로 처리한다.

Extension

- 종류: Literal
- 범위: 31자 이내
- EXT 절 이름이 아닌 다름 값을 확장자로 사용할 경우 설정한다.

3.16.2. MIME-Type

대부분의 알려진 MIME-Type은 기본으로 제공된다.

다음 리스트에 있는 항목들은 EXT 절에 설정할 필요는 없다. 단, 모두 HTML을 처리하는 Worker Thread 처리하므로, 해당 확장자를 다른 서버에서 처리할 경우 EXT 절에 처리할 SvrType으로 설정해야 한다.

Extension	MIME-Type	Extension	MIME-Type
avi	"video/video-x-msvideo"	asc	"text/plain"
au	"audio/basic"	ai	"application/postscript"
aif	"audio/x-aiff"	aiff	"audio/x-aiff"
af	"audio/x-aiff"	aifc	"audio/x-aiff"
bmp	"image/bmp"	bin	"application/octet-stream"
bcpio	"application/x-bcpio"	cpio	"application/x-cpio"
csh	"application/x-csh"	cpt	"application/mac-compactpro"
class	"application/octet-stream"	cdf	"application/x-netcdf"
css	"text/css"	doc	"application/msword"
dvi	"application/x-dvi"	dms	"application/octet-stream"
dcr	"application/x-director"	dir	"application/x-director"
dxr	"application/x-director"	eps	"application/postscript"
exe	"application/octet-stream"	ez	"application/andrew-inset"
etx	"text/x-setext"	gif	"image/gif"
gtar	"application/x-gtar"	html	"text/html"
htm	"text/html"	hqx	"application/mac-binhex40"
hdf	"application/x-hdf"	hwp	"application/x-hwp"
hif	"application/x-hif"	hpt	"application/x-hpt"
hst	"application/x-hst"	ief	"image/ief"
igs	"model/iges"	iges	"model/iges"
ice	"x-conference/x-cooltalk"	jpg	"image/jpeg"
js	"application/x-javascript"	jpeg	"image/jpeg"
jpe	"image/jpeg"	kar	"audio/midi"
latex	"application/x-latex"	lha	"application/octet-stream"
lzh	"application/octet-stream"	mpg	"video/mpeg"
mpeg	"video/mpeg"	mpe	"video/mpeg"
mid	"audio/midi"	midi	"audio/midi"
mov	"video/quicktime"	mif	"application/vnd.mif"
man	"application/x-troff-man"	me	"application/x-troff-me"

Extension	MIME-Type	Extension	MIME-Type
ms	"application/x-troff-ms"	mpga	"audio/mpeg"
mp2	"audio/mpeg"	mp3	"audio/mpeg"
msh	"model/mesh"	mesh	"model/mesh"
movie	"video/video-x-sgi-movie"	nc	"application/x-netcdf"
oda	"application/oda"	pdf	"application/pdf"
ps	"application/postscript"	ppt	"application/vnd.ms-powerpoint"
pgn	"application/x-chess-pgn"	pdb	"chemical/x-pdb"
png	"image/png"	pnm	"image/x-portable-anymap"
pbm	"image/x-portable-bitmap"	pgm	"image/x-portable-graymap"
ppm	"image/x-portable-pixmap"	qt	"video/quicktime"
rtf	"application/rtf"	ra	"audio/x-realaudio"
rgb	"image/x-rgb"	roff	"application/x-troff"
rmm	"audio/x-pn-realaudio"	ram	"audio/x-pn-realaudio"
rm	"application/vnd.rn-realmedia"	rpm	"application/x-rpm"
ras	"image/x-cmu-raster"	rtx	"text/richtext"
rt	"text/vnd.rn-realtext"	rv	"video/vnd.rn-realvideo"
rf	"image/vnd.rn-realflash"	rp	"image/vnd.rn-realpix"
sgm	"text/sgml"	src	"application/x-wais-source"
snd	"audio/basic"	smi	"application/smil"
smil	"application/smil"	spl	"application/x-futuresplash"
skp	"application/x-koan"	skd	"application/x-koan"
skt	"application/x-koan"	skm	"application/x-koan"
sh	"application/x-sh"	shar	"application/x-shar"
swf	"application/x-shockwave-flash"	sit	"application/x-stuffit"
sv4cpio	"application/x-sv4cpio"	sv4crc	"application/x-sv4crc"
silo	"model/mesh"	sgml	"text/sgml"
sdp	"application/sdp"	txt	"text/plain"
tar	"application/x-tar"	tcl	"application/x-tcl"
tex	"application/x-tex"	texinfo	"application/x-texinfo"
texi	"application/x-texinfo"	t	"application/x-troff"
tr	"application/x-troff"	tiff	"image/tiff"
tif	"image/tiff"	tsv	"text/tab-separated-values"
ustar	"application/x-ustar"	vrml	"model/vrml"

Extension	MIME-Type	Extension	MIME-Type
vcd	"application/x-cdlink"	wav	"audio/x-wav"
wrl	"model/vrml"	xls	"application/vnd.ms-excel"
xyz	"chemical/x-pdb"	xbm	"image/x-xbitmap"
xpm	"image/x-pixmap"	xwd	"image/x-windowdump"
xml	"text/xml"	zip	"application/zip"

3.16.3. 예제

다음은 EXT 절을 설정한 예제이다.

```
*EXT
htm
    Mimetype = "text/html",
    SvrType = HTML
php4
    Mimetype = "text/html",
    SvrType = PHP
do
    Mimetype = "text/html",
    SvrType = JSV
```

3.17. SSL 절

WebtoB에서 사용할 SSL의 기능을 설정한다. 해당 절에 정의된 형태로 SSL 서비스를 한다.

3.17.1. 설정 항목

다음은 SSL 절의 환경설정 형식이다.

```
*SSL
name    # String[32]
    CertificateFile = Literal[256],     # $ENV, R.PATH
    CertificateKeyFile = Literal[256],  # $ENV, R.PATH
   #CACertificatePath = Literal[256],   # $ENV, R.PATH
   #CACertificateFile = Literal[256],   # $ENV
   #CertificateChainFile = Literal[256], # $ENV, R.PATH
   #VerifyDepth = Numeric,              # 0 (0-)
   #VerifyClient = Numeric,             # 0 (0-3)
   #FakeBasicAuth = Boolean,            # N
   #Protocols = Literal[256],           # LIST
   #RequiredCiphers = Literal[1024],    # LIST
   #RandomFile = Literal[256],          # $ENV, R.PATH
   #RandomFilePerConnection = Literal[256], # $ENV, R.PATH
   #PassPhraseDialog = Literal[256],    # $ENV, R.PATH
   #CryptoDevice = Literal[256],        # "builtin"
   #RenegotiationLevel = String[256],   # secure (secure, insecure, disable)
   #DHParameter = Literal[256],         # $ENV, R.PATH
```

```
#Options = Literal[256]              # PM.LIST
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

SSL 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 설정한다.

CertificateFile(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- PEM 방식으로 인코딩된 서버의 인증서를 설정한다. 이것은 DER 규칙으로 인코딩되어 있으며 웹에서 전송하기 위해서 ASCII 코드처럼 이용된다. 만일 인증서 역시 암호화된 상태라면 비밀번호(passphrase)를 물어본다.
- 콤마(,)로 구분하여 4개까지 설정할 수 있다.

CertificateKeyFile의 설정 개수와 같아야 한다.

CertificateKeyFile(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 서버에서 쓰이는 PEM 방식으로 인코딩된 인증서의 개인 Key를 설정한다.
- Key가 인증서와 함께 조합되지 않았다면 이 지시자를 이용하여 Key의 위치를 지정해야 한다. 일반적으로 WebtoB의 SSL 디렉터리에 가져다 두게 된다.
- 콤마(,)로 구분하여 4개까지 설정할 수 있다.

CertificateFile의 설정 개수는 같아야 한다.

CACertificatePath

- 종류: Literal
- 범위: 255자 이내
- 인증서를 저장할 디렉터리를 설정한다.

- 인증서는 받아들이기 위해 준비할 사용자의 인증서를 인증할 내용을 담고 있다. 인증서는 보통 PEM 방식으로 인코딩되어 있어야 한다.

CACertificateFile

- 종류: Literal
- 범위: 255자 이내
- 단일 CA(Certificate Agent: 인증을 대신해 주는 기업)로부터의 사용자 인증만 받고 싶다면 CACertificatePath 지시자가 아닌, 이 지시자를 이용하여 단일 PEM으로 인코딩된 인증 파일을 사용해야 한다.

CertificateChainFile

- 종류: Literal
- 범위: 255자 이내
- 서버 인증서(Certificate)의 인증서 연쇄(Certificate Chain)를 구성할 때 필요한 상위 인증기관들(CAs)의 인증서(Certificate) 경로를 설정한다. 단, 클라이언트의 인증(Authentication)을 사용하기 위해서는 CACertificateFile이나 CACertificatePath에 설정해야 한다.

VerifyDepth

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 실제 업무에 적용되는 경우에 인증에서 개입할 부분은 순서대로 다른 CA에 의해서 서로를 인증하는 CA에 관한 것이다. VerifyDepth 항목은 얼마나 깊은 레벨로 연결된 CA들을 추적하여 인증할 것인지를 설정한다. 단 하나의 인증 CA만 필요하다면 1로 설정한다.

VerifyClient

- 종류: Numeric
- 기본값: 0
- 사용자에게 요청할 인증 레벨을 설정한다.
- 다음은 인증 레벨에 대한 설명이다.

레벨	설명
0	아무런 인증 요청을 하지 않는다.
1	사용자는 사용 가능한 인증을 서버에게 보여 주어야 한다.
2	사용 가능한 인증을 반드시 서버에게 보여 주어야 한다.

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

FakeBasicAuth

- 종류: Boolean
- 기본값: N
- 사용자 측의 인증 버전인 한 줄짜리 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. 코드에 미리 정해진 비밀번호를 추가해 둔다.

Protocols

- 종류: Literal
- 범위: 255자 이내
- 기본값: "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3"
- 서버가 사용할 수 있는 프로토콜을 설정한다. 특정 TLS 버전에 대한 지원 여부를 설정할 수 있다. "SSLv2", "SSLv3"는 더이상 지원하지 않는다.
- 특정 프로토콜을 사용하지 않을 경우 프로토콜 이름 앞에 하이픈(-)을 설정한다.
- 동일한 포트를 사용하는 호스트가 여러 개 설정될 경우 가장 먼저 기재된 호스트의 *SSL.Protocols 설정만 적용될 수 있다.

RequiredCiphers

- 종류: Literal
- 범위: 1023자 이내
- 기본값: "HIGH:MEDIUM:!SSLv2:!PSK:!SRP:!ADH:!AECDH:!EXP:!RC4:!IDEA:!3DES"
- 서버가 사용할 수 있는 cipher를 설정한다. 특정 cipher 및 SSL, TLS 버전에 대한 지원 여부를 설정할 수 있다. WebtoB는 OpenSSL을 사용하기 때문에 cipher 이름은 OpenSSL 설명서를 참고한다 .

RandomFile

- 종류: Literal
- 범위: 255자 이내
- SSL에서 이용할 Random Seed를 위한 파일을 선택하는 것이다. 이 파일을 임의로 선택하면, WebtoB는 SSL을 위한 암호를 만들 때 이 파일에서 임의의 값을 추출하여 Random Seed를 만든다.

RandomFilePerConnection

- 종류: Literal
- 범위: 255자 이내
- RandomFile을 설정할 때 실제 각 SSL을 통한 연결을 맺을 때 몇 개의 파일을 이용하여 Random Seed를 만들 것인지 설정한다.

PassPhraseDialog

- 종류: Literal
- 기본값: "builtin"
- SSL을 사용하는 경우 암호화된 개인키(encrypted private key) 파일들에 대한 암호문을 얻기 위한 방식을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	내용
builtin	WebtoB가 기동될 때 암호문을 입력할 것을 요구한다.
exec:<program path>	WebtoB가 기동될 때 해당 프로그램을 실행하고, 그 출력 결과를 암호문으로 사용한다. exec로 실행되는 파일은 컴파일된 실행 파일이나 쉘 스크립트가 이용될 수 있다.
file:<passphrase file path>	WebtoB가 기동될 때 wsmkppd 툴을 이용하여 생성된 passphrase file을 이용하여 암호문을 사용한다.

CryptoDevice

- 종류: Literal
- 범위: 255자 이내
- 기본값: "builtin"(OpenSSL에서 구현한 방식을 사용한다.)
- 외부 암호화 장비를 사용하는 경우 설정한다.

RenegotiationLevel

- 종류: String
- 범위: 255자 이내
- 기본값: secure
- SSL을 사용하는 경우 재협상(Renetotiation)의 레벨(Level)을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	내용
secure	클라이언트와 웹 서버가 모두 안전한 경우 재협상을 진행한다. (예: RFC5746)
insecure	클라이언트와 웹 서버가 안전하지 않더라도 재협상을 진행한다. (예: CVE-2009-3555)
disable	어떠한 경우에도 재협상을 진행하지 않는다.

안전하지 않은 경우 재협상이 진행되면 MITM(Man in the Middle) Attack 또는 DoS(Denial of Service) Attack에 취약할 수 있기 때문에 주의해야 한다.

DHParameter

- 종류: Literal
- 범위: 255자 이내
- 암호화 통신 보안 강화를 위해 새롭게 생성한 고유의 Diffie-Hellman group을 사용할 수 있도록 설정한다.

  다음은 Diffie-Hellman group 생성 방법이다. (2048bit)

```
wbssl dhparam -out dhparams.pem 2048
```

SSLServerCipherPref

- 종류: Boolean
- 기본값: N
- SSL 알고리즘을 선택하는 경우 RequiredCiphers에 등록된 순서대로 사용할지 여부를 결정한다.

Options

- 종류: Literal
- 범위: 255자 이내
- 다음은 설정 가능한 옵션에 대한 설명이다.

옵션	내용

StdEnvVars	SSL Handshake 과정에 사용된 기본 정보를 CGI/PHP, JEUS, Reverse Proxy 내부 서버에서 사용 가능하도록 SSL 환경변수 또는 HTTP Request Header 값을 생성한다.
	• CGI/PHP 서버에서 사용 가능한 환경변수: SSL_SESSION_ID(SSL session id), SSL_CIPHER(사용된 cipher), SSL_CIPHER_USEKEYSIZE(사용된 cipher bit)이다.
	• JEUS 또는 Reverse Proxy 내부 서버에서 사용 가능한 HTTP Request Header: WJP-SSL-SESSION-ID(SSL session id), WJP-SSL-CIPHER(사용된 cipher), WJP-SSL-CIPHER-USEKEYSIZE(사용된 cipher bit)이다.
	다음은 각 언어별 예이다.
	• CGI
	`c - getenv("SSL_CIPHER") perl - $ENV{'SSL_CIPHER'}`
	• PHP
	`$_SERVER['SSL_CIPHER']`
	• JSP
	`request.getHeader("WJP-SSL-CIPHER")`

옵션	내용

ExportClientCert	클라이언트 인증을 실행하는 경우(VerifyClient = 2 설정을 적용한 경우) 클라이언트 인증서를 CGI/PHP, JEUS, Reverse Proxy 내부 서버에서 사용 가능하도록 SSL 환경변수 또는 HTTP Request Header 값을 생성한다. 클라이언트 인증서의 확장필드(Extension Fields - Key Usage, Certificate Policies, CRL Distribution Points 등)를 처리하기 위해서는 해당 설정으로 클라이언트 인증서 정보를 얻어 파싱 후 사용해야 한다.
	• CGI/PHP 서버에서 사용 가능한 환경변수: SSL_CLIENT_CERT(클라이언트 인증서-PEM 인코딩된 String 값)이다.
	• JEUS 서버 또는 Reverse Proxy 내부 서버에서 사용가능한 HTTP Request Header: WJP-SSL-CLIENT-CERT(클라이언트 인증서-PEM 인코딩된 String 값)이다.
	다음은 각 언어별 예이다.
	• CGI
	`c - getenv("SSL_CLIENT_CERT") perl - $ENV{'SSL_CLIENT_CERT'}`
	• PHP
	`$_SERVER['SSL_CLIENT_CERT']`
	• JSP
	`request.getHeader("WJP-SSL-CLIENT-CERT")`

3.17.2. CA 명령어 사용

UNIX에서 사용하는 CA 명령어에 대해서 설명한다.

CertificateFile 또는 CertificateKeyFile 등의 항목을 통하여 사용자 인증을 위한 Key 및 인증서가 있는 파일의 위치를 설정할 수 있다. 보통 외부 보안 업체에서 이 Key를 발급 받아서 이용하는 경우도 있으나, 서버의 관리자가 이를 직접 만들어서 접속하는 사용자와 보안에 관한 정보를 직접 주고받는 것도 가능하다. 이때 서버는 CA라는 프로그램을 사용해서 보안에 대한 Key 및 인증서를 생성한다.

CA 프로그램은 WebtoB의 실행 디렉터리(/bin) 아래에 존재한다.

- 인증서 생성

  실제 서버가 인증서를 만드는 작업으로 CA 명령어를 사용해서 사용자에게 인증에 필요한 정보를 내보낸다. 명령어를 통하여 새로운 인증서를 만들어 내는 경우 개인 Key와 인증서가 생성된다. 인증서를 생성하는 경우 비밀번호를 입력하는데, WebtoB를 기동시킬 때 비밀번호를 물어본다. 인증서를 생성할 때 입력한 비밀번호와 동일한 비밀번호를 입력해야 WebtoB의 기동이 가능하다. 따라서 입력한 비밀번호는 반드시 기억해야 한다.

```
$ CA -newcert
```

- 인증서 확인

  입력으로 들어온 파일에 대하여 정당한 것인지 확인한다.

```
$ CA -newca
```

- Certificate Request Form 생성

  특정한 Request Form를 생성하면 Verisign 등에서 인증서를 제작하거나 인증을 하는 기관에서 필요로 하게 된다. 외부 인증 기관에 필요한 인증 Request Form을 만들어 낸다.

```
$ CA -newreq
```

- Sign 기입

  만들어진 Key에 Sign을 입력한다.

```
$ CA -sign
```

기타 SSL에 관련된 정보를 알고 싶다면 다른 Reference나 관련 사이트를 참고한다. SSL 자체가 상당히 복잡하고 많은 내용을 담고 있기 때문에 이를 자세하게 설명하는 것은 불가능하므로, 가급적 관련 자료를 보기를 권한다.

3.17.3. 예제

다음은 SSL 절을 설정한 예제이다.

```
*SSL
ssl1
    CertificateFile = "/user/webtob/ssl/newcert.pem",
    CertificateKeyFile = "/user/webtob/ssl/newcert.pem",
    PassPhraseDialog = "exec: /user/webtob/ssl/pass.sh"
```

3.18. PROXY_SSL 절

WebtoB가 Proxy 역할을 할 때 사용할 SSL의 기능을 설정한다. 해당 절에 정의된 형태로 SSL 서비스한다.

3.18.1. 설정 항목

다음은 PROXY_SSL 절의 환경설정 형식이다.

```
*PROXY_SSL
name    # String[32]
    Verify = Numeric,                   # 0 (0-3)
   #VerifyDepth = Numeric,              # 0 (0-)
```

```
#CACertificatePath = Literal[256],   # $ENV, R.PATH
#CACertificateFile = Literal[256],   # $ENV
#CheckPeerValidPeriod = Boolean,     # N
#Protocols = Literal[256],           # LIST
#RequiredCiphers = Literal[1024],    # LIST
#CertificateFile = Literal[256],     # $ENV, R.PATH
#CertificateKeyFile = Literal[256],  # $ENV, R.PATH
#CertificateChainFile = Literal[256], # $ENV, R.PATH
#PassPhraseDialog = Literal[256]     # $ENV, R.PATH
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

PROXY_SSL 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 설정한다.

Verify

- 종류: Numeric
- 기본값: 0
- 내부 서버의 인증서에 대하여 인증 타입을 설정한다.
- 다음은 인증 타입에 대한 설명이다.

레벨	설명
0	내부 서버의 인증서 인증 과정을 진행하지 않는다.
1	내부 서버는 사용 가능한 인증을 보여 주어야 하며 WebtoB가 내부 서버의 인증서를 받은 경우 인증서 인증 과정을 진행한다.
2	내부 서버는 사용 가능한 인증을 반드시 보여 주어야 하며 WebtoB가 내부 서버의 인증서 인증 과정을 진행한다.
3	내부 서버는 사용 가능한 인증을 보여 주어야 하며 WebtoB가 인증서를 가지고 있지 않은 상황에서는 내부 서버의 인증서 인증 과정을 진행하지 않는다.

VerifyDepth

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0

- 실제 업무에 적용되는 경우에 인증에서 개입할 부분은 순서대로 다른 CA에 의해서 서로를 인증하는 CA에 관한 것이다. VerifyDepth 항목은 얼마나 깊은 레벨로 연결된 CA들을 추적하여 인증할 것인지를 설정한다. 단 하나의 인증 CA만 필요하다면 1로 설정한다.

CACertificatePath

- 종류: Literal
- 범위: 255자 이내
- 인증서를 저장할 디렉터리를 설정한다.
- 인증서는 연결하기 위해 준비할 서버의 인증서를 인증할 내용을 담고 있다. 인증서는 보통 PEM 방식으로 인코딩되어 있어야 한다.

CACertificateFile

- 종류: Literal
- 범위: 255자 이내
- 단일 CA(Certificate Agent: 인증을 대신해 주는 기업)로부터의 서버 인증만 받고 싶다면 CACertificatePath 지시자가 아닌 이 지시자를 이용하여 단일 PEM으로 인코딩된 인증 파일을 사용해야 한다.

CheckPeerValidPeriod

- 종류: Literal
- 기본값: N
- 내부 서버의 인증서 인증과정을 진행하는 경우 인증서의 유효기간을 검증할 지 여부를 설정한다 .

Protocols

- 종류: Literal
- 범위: 255자 이내
- 기본값: "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3"
- 클라이언트가 사용할 수 있는 프로토콜을 설정한다. 특정 SSL, TLS 버전에 대한 지원 여부를 설정할 수 있다. "TLSv1.1, TLSv1.2"는 WBSSL 1.0.1 이상인 경우에 지원한다. "SSLv2, SSLv3"는 더이상 지원하지 않는다.
- 특정 프로토콜을 사용하지 않을 경우 프로토콜 이름 앞에 하이픈(-)을 설정한다.

RequiredCiphers

- 종류: Literal

- 범위: 1023자 이내
- 기본값: "HIGH:MEDIUM:!SSLv2:!PSK:!SRP:!ADH:!AECDH:!EXP:!RC4:!IDEA:!3DES"
- 클라이언트가 사용할 수 있는 cipher를 설정한다. 특정 cipher 및 SSL, TLS 버전의 지원 여부를 설정할 수 있다. WebtoB는 OpenSSL을 사용하기때문에, cipher 이름은 OpenSSL 설명서를 참고한다 .

CertificateFile

- 종류: Literal
- 범위: 255자 이내
- PEM 방식으로 인코딩된 클라이언트 인증서를 설정한다. 내부 서버가 클라이언트 인증을 원하는 경우 반드시 설정해야 한다. 이것은 DER 규칙으로 인코딩되어 있으며 웹에서 전송하기 위해서 ASCII 코드처럼 이용된다. 만일 인증서 역시 암호화된 상태라면 비밀번호(passphrase)를 물어본다.

CertificateKeyFile

- 종류: Literal
- 범위: 255자 이내
- 클라이언트로 쓰이는 PEM 방식으로 인코딩된 인증서의 개인 Key를 설정한다. 내부 서버가 클라이언트 인증을 원하는 경우 반드시 설정해야 한다.
- Key가 인증서와 함께 조합되지 않았다면 이 지시자를 이용하여 Key의 위치를 지정해야 한다. 일반적으로 WebtoB의 SSL 디렉터리에 가져다 두게 된다.

CertificateChainFile

- 종류: Literal
- 범위: 255자 이내
- 클라이언트 인증서(Certificate)의 인증서 연쇄(Certificate Chain)를 구성할 때 필요한 상위 인증기관들(CAs)의 인증서(Certificate) 경로를 설정한다.

PassPhraseDialog

- 종류: Literal
- 기본값: "builtin"
- PROXY_SSL에서 클라이언트 인증(CertificateFile, CertificateKeyFile)을 사용하는 경우 암호화된 개인키(encrypted private key) 파일들에 대한 암호문을 얻기 위한 방식을 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
builtin	WebtoB가 기동될 때 암호문을 입력할 것을 요구한다.
exec:<program path>	WebtoB가 기동될 때 해당 프로그램을 실행하고, 그 출력 결과를 암호문으로 사용한다. exec로 실행되는 파일은 컴파일된 실행 파일이나 셸스크립트가 이용될 수 있다.
file:<passphrase file path>	WebtoB가 기동될 때 wsmkppd 툴을 이용하여 생성된 passphrase file을 이용하여 암호문을 사용한다.

3.18.2. 예제

다음은 SSL 절을 설정한 예제이다.

```
*PROXY_SSL
proxy_ssl1
    Verify = 2,      # 내부 서버 인증서 인증절차 진행
    VerifyDepth = 1,
    CACertificateFile="/user/webtob/ssl/server.crt",
    CheckPeerValidPeriod = Y
```

3.19. ERRORDOCUMENT 절

WebtoB에서 에러 문제가 발생했을 때 다음과 같은 4가지 방법으로 대응할 수 있다.

1. 소스코드에 정의된 에러 메시지를 출력한다.
2. 사용자가 정의한 에러 메시지를 출력한다.
3. 로컬 URL로 재전송한다.
4. 외부 URL로 재전송한다.

2번째, 3번째, 4번째의 경우엔 ErrorDocument Section을 설정하여 특정 HTTP Response Status Code 값에 대한 특정 페이지로 Redirect를 시켜준다. HTTP 401 Status Code를 제외한 HTTP Status Code를 모두 설정할 수 있다.

3.19.1. 설정 항목

다음은 ERRORDOCUMENT 절의 환경설정 형식이다.

```
*ERRORDOCUMENT
name    # String[32]
    Status = Numeric,                   # (HTTP status code)
    #MultiStatus = Literal[256],         # (HTTP status code)
    Url = Literal[256]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

ERRORDOCUMENT 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

Status(필수 항목)

- 종류: Numeric
- HTTP Status Code 값을 설정한다.

MultiStatus

- 종류: Literal
- 범위: 255자 이내
- 동일한 url에 대해 여러 개의 HTTP Status Code 값을 설정한다.
- Status와 MultiStaus 중 반드시 한 개 이상 설정해야 하며 동시에 설정할 수도 있다. 동시에 설정한 경우 명시된 순서대로 적용된다.

Url(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- DocRoot 이하의 상대 경로가 되거나 클라이언트가 해석할 수 있는 전체 경로 값을 설정한다.

3.19.2. 예제

다음은 ERRORDOCUMENT 절을 설정한 예제이다.

```
*ERRORDOCUMENT
forbidden403
    Status = 403,
    Url = "err/403.html"
notfound404
    Status = 404,
    Url = "http://www.tmax.co.kr/404.html"
```

3.20. EXPIRES 절

EXPIRES 절은 클라이언트 요청에 따라 전송되는 서버 Response Header의 정보를 설정한다.

특정 MIME-Type 문서를 전송할 때 서버 Response Header 내에 전송되는 문서의 Expiry Date를 설정한다. 클라이언트의 브라우저 프로그램이 같은 웹 사이트에 재접속했을 때 이미 캐시에 해당 사이트의 웹 문서가 저장되어 있다면 수정된 문서만을 받으려고 할것이다.

클라이언트의 브라우저는 웹 서버에게 특정 MIME-Type과 해당 MIME의 만료일자를 보내면서 Expiry Date가 경과된 문서만을 재전송해 달라고 웹 서버에게 요청한다. 웹 서버는 클라이언트의 Request Header에 포함된 MIME-Type과 Expiry Date를 분석하고 해당 MIME-Type 문서가 현재 일자보다 기간이 경과되었을 때는 재전송한다.

클라이언트가 웹 서버에게 보내는 Expiry Date는 웹 서버가 설정 파일을 참조해서 클라이언트에게 전송한다. EXPIRES 절은 Expiry Date의 설정과 관계되는 것을 정의한다.

3.20.1. 설정 항목

다음은 EXPIRES 절의 환경설정 형식이다.

```
*EXPIRES
name    # String[32]
    ExpiresTime = Literal[256],
   #Mimetype = Literal[128],
   #Url = Literal[256]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

EXPIRES 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

ExpiresTime(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 아래와 같이 2가지 방법으로 Expiry Date를 설정한다. ExpiresTime만 설정한 경우 모든 요청에 대해 Expiry Date를 Response Header에 설정한다.
    - <code><seconds>

항 목	설명
<code>	◦ 'M': file Modification date + <seconds>로 Expiry Date를 설정한다. ◦ 'A': Access time + <seconds>로 Expiry Date를 설정한다.
<second>	일반적으로 사용되는 시간 주기(seconds)를 설정한다. ◦ 1 hour(3600) ◦ 1day(86400) ◦ 1week(604800) ◦ 1moth(241920)

◦ <base> [plus] {<num> <type>}*

항 목	설명
<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’

Mimetype

- 종류: Literal
- 범위: 127자 이내
- 응답의 MIME-Type에 따라 Expiry Date를 설정할 수 있다.

Url

- 종류: Literal
- 범위: 255자 이내
- Request URL이 일치하면 해당 Expiry Date를 설정할 수 있다.

3.20.2. 예제

다음은 EXPIRES 절을 설정한 예제이다.

```
*EXPIRES
exp1
    MimeType = "text/html",
    ExpiresTime = "A604800"
exp2
    MimeType = "image/gif",
    ExpiresTime = "A2419200"
exp3
    ExpiresTime = "A86400"
exp4
    Url = "/news/",
    ExpiresTime = "modification 1 days"
exp5
    MimeType = "text/html",
    ExpiresTime = "access plus 1 weeks"
exp6
    ExpiresTime = "M86400"
exp7
    Url = "/image/",
    MimeType = "image/gif",
    ExpiresTime = "access plus 1 months"
exp8
    MimeType = "text/html",
    ExpiresTime = "M86400"
```

3.21. TCPGW 절

TCPGW 절은 특정 Port나 IP 주소로 들어오는 TCP 연결을 다른 서버로 전송하는 중계(proxy)하는 역할을 한다. TCPGW는 전송되는 데이터를 전혀 해석하지 않는다. AccessLog는 남기지 않으며, 클라이언트가 접속하거나 접속을 종료할 때 SysLog를 남긴다.

3.21.1. 설정 항목

다음은 TCPGW 절의 환경설정 형식이다.

```
*TCPGW
name    # String[32]
    ServerAddress = Literal[1024],      # LIST[100]
```

```
#Port = Literal[1024],              # (1-65535), LIST[100]
#Listen = Literal[1024],            # LIST[100]
#SSLFlag = Boolean,                 # N
#SSLName = String[32],
#Timeout = Numeric,                 # 300 (0-)
#CheckAliveTime = Numeric,          # 0 (0-)
#ConnectTimeout = Numeric,          # 5 (0-)
#ConnectDirection = String[256],    # "CS"
#Schedule = String[256],            # "RR"
#AccessName = String[32],
#ClientConnectionLog = Boolean      # Y
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

TCPGW 절 이름

- 종류: String
- 범위: 31자 이내
- 절 이름을 정의한다.

ServerAddress(필수 항목)

- 종류: Literal
- 범위: 1023자 이내
- 클라이언트의 요청을 처리할 서버들의 IP 주소와 포트 번호를 설정한다.
- 여러 개의 서버 IP 주소와 포트 번호를 설정하는 것도 가능하며, 현재는 최대 100개까지의 서버 IP 주소와 포트 번호를 동시에 설정하는 것이 가능하다. 현재 설정된 서버들 간의 분배규칙은 Round Robin만 지원한다.

Port

- 종류: Literal
- 범위: 1023자 이내
- 웹 서버에서 클라이언트의 요청을 대기하는 포트 번호를 설정한다.
- 여러 Port를 동시에 정의하는 것도 가능하며, 현재는 최대 100개까지의 포트 번호를 동시에 설정할 수 있다. Port의 설정을 통해서 여러 개의 포트 번호가 동시에 대기하는 것이 가능하다.

Port와 Listen 항목이 같이 설정된 경우 Port 설정은 무시된다.

Listen

- 종류: Literal
- 범위: 1023자 이내
- 웹 서버에서 클라이언트의 요청을 대기하는 IP 주소와 포트 번호를 설정한다.
- 여러 개의 IP 주소와 포트 번호를 설정하는 것도 가능하며, 현재는 최대 100개까지의 IP 주소와 포트 번호를 동시에 설정하는 것이 가능하다. 여러 개의 IP 주소를 가진 서버인 경우 특정 IP 주소를 통한 요청만 받아들이기 위해서 사용할 수 있다.

SSLFlag

- 종류: Boolean
- 기본값: N
- SSL을 적용할 경우 설정한다.

SSLName

- 종류: String
- 범위: 31자 이내
- 사용할 SSL 절 이름을 설정한다.

Timeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 300
- 사용자가 요청 후 해당 서버로부터 지정된 시간 동안 응답이 없는 경우 접속을 종료하기 위한 시간을 설정한다.
- 해당 시간을 0으로 설정하게 되면, TimeOut을 적용하지 않겠다는 의미이다.

CheckAliveTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0
- ServerAddress에 설정한 요청을 분배할 서버 중 서버에 장애가 발생한 경우 매 요청할 때마다 그 상태를 확인하지 않고, 최초 한 번만 접속하여 장애가 확인되면 다음 요청할 때에는 분배에서 제외한다.

- 장애가 발생한 서버를 CheckAliveTime 간격으로 상태를 확인하여 다음 요청할 때 분배에 포함시킬 것인지 아닌지를 결정한다.
- CheckAliveTime를 0으로 지정한 경우 기존의 방식처럼 매 요청할 때마다 서버 상태를 확인하게 된다.

ConnectTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 5
- 클라이언트의 요청을 WebtoB TCP 게이트웨이가 받아서 설정된 상대 서버로 접속을 시도할 때 일정시간 동안 접속 요구에 대한 응답이 없으면 다른 서버로 재분배되도록 시간을 설정한다.
- 해당 시간을 0으로 설정하게 되면, ConnectTimeOut을 적용하지 않겠다는 의미이다.

ConnectDirection

- 종류: String
- 범위: CS | SC
- 기본값: CS
- 연결 방향이 Client to Server(CS)인지, Server to Client(SC)인지 설정한다.

Schedule

- 종류: String
- 범위: RR | FA
- 기본값: RR
- 외부 서버를 여러 개 설정하였을 경우 요청을 분배하는 방식을 설정한다.

AccessName

- 종류: String
- 범위: 31자 이내
- ACCESS 절의 AccessName에 설정된 규칙에 따라 사용자 접속의 허용 여부를 설정한다.

ClientConnectionLog

- 종류: Boolean
- 기본값: Y
- 클라이언트가 TCP 게이트웨이 포트에 연결을 맺거나 이미 맺어진 연결을 종료하는 경우 해당 정보를 SysLog에 남길지 여부를 설정한다 .

3.21.2. 예제

다음은 TCPGW 절을 설정한 예제이다.

```
*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
tcpgw3
    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"
```

3.22. REVERSE_PROXY_GROUP 절

REVERSE_PROXY_GROUP 절은 REVERSE_PROXY 절 설정을 그룹으로 관리할 필요가 있을 때 사용하며, 다중 서버를 구성할 수 있다. REVERSE_PROXY(내부 서버)를 여러 개 설정하여 그룹으로 묶어 로드 밸런싱 및 Sticky Session routing을 할 수 있으며, WAS 연동에 사용할 수 있다.

3.22.1. 설정 항목

다음은 REVERSE_PROXY_GROUP 절의 환경설정 형식이다.

```
*REVERSE_PROXY_GROUP
name    # String[32]
    PathPrefix = Literal[256],
    ServerPathPrefix = Literal[256],
   #RegExp = Literal[512],
   #SetHostHeader = Literal[256],
   #VhostName = String[1024],                # LIST[64], MULTI
   #RewriteRedirect = Literal[256],          # MULTI[16]
```

```
#RewriteCookieDomain = Literal[256],
#RewriteCookiePath = Literal[256],
#RewriteHtmlMaxSize = Numeric,              # 10240 (1-)
#RewriteHtmlUrl = Literal[256],             # MULTI[32]
#HtmlUrl = Literal[256],                    # MULTI[64]
#WBRoutingCookieKey = Literal[256],
#SessionIdCookieKey = Literal[256],
#FlexibleStickySessionRouting = Boolean,    # N
#LBBackup = Literal[256],
#FailbackCheckTime = Numeric,               # 60 (0-)
#AccessName = String[32],
#Headers = Literal[256]                     #LIST[15]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

REVERSE_PROXY_GROUP 절 이름

- 종류: String
- 범위: 31자 이내
- Reverse Proxy Group 이름을 설정한다.

PathPrefix(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- HTTP Request URL이 지정된 값으로 시작하면 해당 요청을 Reverse Proxy로 처리한다.

```
PathPrefix = "/internal/"
```

ServerPathPrefix(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- HTTP Request URL의 PathPrefix에 해당하는 부분이 ServerPathPrefix에 설정된 값으로 교체된 후 수정된 요청이 내부 서버(ServerAddress)로 전달된다.
- 다음의 설정으로 HTTP Request URL이 "/internal/abc.html"이 "/docs/abc.html"로 변경된 후 요청이 내부 서버로 전송된다.

```
ServerPathPrefix = "/docs/"
```

RegExp

- 종류: Literal
- 범위: 511자 이내
- HTTP Request URL이 Regular expression과 매칭되면 해당 요청을 Reverse Proxy로 처리한다.
- '!'로 시작하는 경우 exclude를 의미한다.

```
RegExp = "\.(do|jsp)$"
```

SetHostHeader

- 종류: Literal
- 범위: 255자 이내
- Reverse Proxy를 사용하여 내부 서버로 요청을 포워딩할 때 Host Header의 값을 설정한다. 설정하지 않은 경우 Host Header의 값은 ServerAddress에 설정한 값이 설정된다.

  "$BypassHostHeader"로 설정한 경우 클라이언트가 보낸 Host Header의 값을 그대로 사용한다.

```
SetHostHeader = "reverseproxy.server.com"
```

VhostName

- 종류: String
- 범위: 1023자 이내
- Reverse Proxy가 속하는 VHOST 절에 설정된 Virtual Host의 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

RewriteRedirect

- 종류: Literal
- 범위: 255자 이내
- 내부 서버 응답의 Status Code가 301, 302, 303, 307일 경우 (redirect), Location, Content-Location header field의 absolute URL을 수정한다.
- RewriteRedirect는 흰 공백(whitespace)으로 2개의 string으로 나누어진다.

  첫 번째는 원본 응답의 Location, Content-Location의 URL과 비교된다. URL이 정해진 string으로 시작하면 URL이 수정된다. URL Host 부분은 요청에 사용된 Host, Port로 전환되고, URL Path 부분은 두 번째

string으로 교체된다.

- 다음의 설정으로 원본 응답의 Location인 "http://internal.server.com:80/docs/abc.html"이 "http://webtob:8100/internal/abc.html"로 변경된다.

```
RewriteRedirect = "http://internal.server.com:80/docs/ /internal/"
RewriteRedirect = "http://internal.server.com:80/docs_kr/ /internal_kr/"
RewriteRedirect = "http://internal.server.com:80/docs_ch/ /internal_ch/"
```

- RewriteRedirect 설정은 최대 16개까지 설정할 수 있다.

RewriteCookieDomain

- 종류: Literal
- 범위: 255자 이내
- 내부 서버 응답의 Cookie Header 필드 값의 "domain=" 값을 수정한다. "domain="에 지정된 domain string이 RewriteCookieDomain에 지정된 string과 일치할 경우 "domain=" 값을 사용자 요청의 domain으로 교체한다.
- 다음과 같이 설정하면 WebtoB의 domain이 webtob인 경우 원본 응답의 Cookie인 "jsessionid=abc, domain=internal.server.com"는 "jsessionid=abc, domain=webtob"로 변경된다.

```
RewriteCookieDomain = "internal.server.com"
```

RewriteCookiePath

- 종류: Literal
- 범위: 255자 이내
- 내부 서버 응답의 Cookie Header 필드 값의 "path=" 값을 수정한다. "path="에 지정된 string이 RewriteCookiePath 에 지정된 첫 번째 string으로 시작할 경우 두 번째 string으로 교체한다.
- 다음과 같이 설정하면 원본 응답의 Cookie의 "jsessionid=abc, path=/jeus/application"이 "jsessionid=abc, path=/jeus_proxy/application"으로 변경된다.

```
RewriteCookiePath = "/jeus /jeus_proxy"
```

RewriteHtmlMaxSize

- 종류: Numeric
- 단위: bytes
- 범위: 1 ~ INT_MAX

- 기본값: 10240
- 응답이 HTML 페이지인 경우(Content-Type: text/html), 페이지 내부의 특정 태그 값들을 수정할 수 있는데, 이때 사용되는 내부 버퍼의 최대 크기를 설정한다.
- 응답이 설정된 값보다 크면, 수정되지 않은 원본 응답이 클라이언트로 전송된다.

RewriteHtmlUrl

- 종류: Literal
- 범위: 255자 이내
- HTML 페이지에 포함된 URL들을 변경할 때 사용된다.
- URL이 설정된 첫 번째 string으로 시작하면, 해당하는 부분이 두 번째 string으로 교체된다. URL이 Host를 포함할 경우 Host는 요청에 사용된 WebtoB 서버 주소로 교체된다.
- URL을 포함하는 tag, attribute은 HtmlUrl을 사용하여 정의한다.
- 다음 Meta tag의 content는 예외로 항상 URL을 포함한다고 가정한다.

```
<meta ... http-equiv="refresh" ... content="http://internal.server.com:80/docs/link3.html" ...>
```

```
RewriteHtmlUrl = "http://test2:80/ /proxy/",
RewriteHtmlUrl = "/ /proxy/"
```

- RewriteRedirect 설정은 최대 32개까지 설정할 수 있다.

HtmlUrl

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 HtmlUrl 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- HTML 문서 내부의 URL을 포함하는 tag, attribute 이름을 정한다. 첫 번째 string은 tag 이름이고, 두 번째부터는 attribute 이름을 설정한다.
- 하나 이상의 tag를 설정하려면, HtmlUrl을 여러 번 설정하여 최대 64개까지 설정할 수 있다.

```
HtmlUrl = "a href",
HtmlUrl = "img src longdesc usemap"
```

WBRoutingCookieKey

- 종류: Literal

- 범위: 255자 이내
- 그룹으로 설정된 Reverse Proxy 서버 사이에 자체 routing을 하기 위한 용도로 사용되는 HTTP Cookie의 Key 이름을 설정한다.
- SessionIdCookieKey 설정보다 우선하며, 해당 설정을 하는 경우 REVERSE_PROXY 절에 설정한 StickySessionRoutingId 값은 무시된다.
- Routing id는 Reverse Proxy Group Name과 Reverse Proxy Name이 Base64로 인코딩된 값이 사용된다.
- 아래와 같이 설정한 경우 Response의 Set-Cookie Header에 W2BRID=RPG1.rproxy1 형식으로 추가하여 이후 요청의 Cookie Header를 확인하여 routing에 사용하게 된다.

```
WBRoutingCookieKey = "W2BRID"
```

- WBRoutingCookieKey의 값을 SessionIdCookieKey의 값과 동일하게 설정할 수 있다. 이 경우 Reverse Proxy 내부 서버로부터 받은 응답(Set-Cookie) 헤더에 설정된 key(예: JSESSIONID)가 있는 경우 해당 key의 value에 자체 routingid를 추가하여 클라이언트에게 전달함으로써 이후 요청에 대한 routingid로 사용한다.

SessionIdCookieKey

- 종류: Literal
- 범위: 255자 이내
- 기본값: "JSESSIONID"
- Session routing용으로 사용되는 HTTP Cookie의 Key 이름을 설정한다.

FlexibleStickySessionRouting

- 종류: Boolean
- 기본값: N
- Session routing을 할 때 flexible하게 routing할 것인지를 설정한다.
- 다음은 설정값에 대한 설명이다.

설정값	설명
Y	StickySessionRoutingId를 기준으로 Session routing을 할 때 내부 서버의 MaxPersistentServerConnections가 모두 RUN 상태이면 큐잉(Queuing)하지 않고 다른 StickySessionRoutingId를 가진 내부 서버의 커넥션으로 routing 한다.
N(사용자가 설정하지 않은 경우)	기본으로 Sticky Session routing이 된다. 이는 동일한 StickySessionRoutingId를 가진 내부 서버의 커넥션으로만 routing하고, 해당 내부 서버의 MaxPersistentServerConnections가 모두 RUN 상태이면 큐잉한다는 의미이다.

Flexible routing을 하는 경우 동일한 JSESSIONID를 가진 다른 클라이언트의 요청이 각각

> 다른 서버로 routing될 수 있으므로 주의해야 한다. 기본값 사용을 권장한다.

LBBackup

- 종류: Literal
- 범위: 31자 이내
- 다른 Reverse Proxy 서버들이 모두 READY 상태가 아닐 경우 백업 용도로 사용할 서버를 설정한다.

FailbackCheckTime

- 종류: Literal
- 범위: 0 ~ INT_MAX
- 기본값: 60
- Failover되어 백업 서버가 사용되고 있을 때 Failback하기 위한 시간을 설정한다. 0일 경우 Failback하지 않고, 백업 서버를 계속 사용한다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP 주소에서 들어온 Reverse Proxy 요청에 대하여 허용 또는 거부하는 ACCESS 절 이름을 설정한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

3.22.2. 예제

다음은 REVERSE_PROXY_GROUP 절을 설정한 예제이다.

```
*REVERSE_PROXY_GROUP
rpg_internal
    VhostName = "vhost1",
    PathPrefix = "/internal/",
    RegExp = "\.(do|jsp)$",
    ServerPathPrefix = "/docs/",
```

```
    RewriteRedirect = "http://internal.server.com:80/docs/ /internal/",
    RewriteCookieDomain = "internal.server.com",
    RewriteCookiePath = "/jeus /jeus_proxy",
    RewriteHtmlUrl = "http://internal.server.com:80/ /internal/",
    RewriteHtmlUrl = "/ /internal/",
    HtmlUrl = "a href",
    HtmlUrl = "img src longdesc usemap",
    RewriteHtmlMaxSize = 4194304,
    SessionIdCookieKey = "JSESSIONID",
    FlexibleStickySessionRouting = N
```

3.23. REVERSE_PROXY 절

REVERSE_PROXY 절은 HTTP Proxy 일종으로 외부에서 접근 불가능한 내부 서버로 요청을 전달할 때 설정한다. 클라이언트로부터 온 HTTP 요청을 내부의 서버로 전달하고, 내부 서버로부터 온 HTTP 응답을 클라이언트로 전달한다. 추가적으로 HTTP 요청과 응답을 설정에 따라 조작할 수 있다.

3.23.1. 설정 항목

다음은 REVERSE_PROXY 절의 환경설정 형식이다.

```
*REVERSE_PROXY
name    # String[32]
    PathPrefix = Literal[256],
    ServerPathPrefix = Literal[256],
    ServerAddress = Literal[256],
   #ReverseProxyGroupName = Literal[32],
   #RegExp = Literal[512],
   #SetHostHeader = Literal[256],
   #VhostName = String[1024],                # LIST[64], MULTI
   #HttpInbufSize = Numeric,                 # 16384 (0-)
   #RewriteRedirect = Literal[256],          # MULTI[16]
   #RewriteCookieDomain = Literal[256],
   #RewriteCookiePath = Literal[256],
   #RewriteHtmlMaxSize = Numeric,            # 10240 (1-)
   #RewriteHtmlUrl = Literal[256],           # MULTI[32]
   #HtmlUrl = Literal[256],                  # MULTI[64]
   #StickySessionRoutingId = Literal[256],
   #MinPersistentServerConnections = Numeric, # 0 (0-)
   #MaxPersistentServerConnections = Numeric, # 0 (0-)
   #PersistentServerCheckTime = Numeric,     # 30 (0-)
   #PersistentServerTimeout = Numeric,       # 300 (0-)
   #PersistentServerCheckUrl = Literal[256],
   #ProxySslFlag = Boolean,                  # N
   #ProxySslName = String[32],
   #MaxWebSocketConnections = Numeric,       # 0 (0-)
   #WebSocketSessionTimeout = Numeric,       # 0 (0-)
   #Options = Literal[256],                  # PM.LIST
   #ConnectRetryCount = Numeric,             # 3 (0-)
   #ConnectTimeout = Numeric,                # 5 (0-)
   #AccessName = String[32],
   #Headers = Literal[256],                  # LIST[15]
   #Compression = Literal[256],              # LIST[32], MULTI
```

```
#CompressionMinSize = Numeric              # 1 (1-)
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

REVERSE_PROXY 절 이름

- 종류: String
- 범위: 31자 이내
- Reverse Proxy 이름을 설정한다.

PathPrefix(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 PathPrefix 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다. ReverseProxyGroupName 설정을 사용하는 경우 PathPrefix 설정은 REVERSE_PROXY_GROUP 절에 설정한 값과 동일해야 한다.
- HTTP Request URL이 지정된 값으로 시작하면 해당 요청을 Reverse Proxy로 처리한다.

```
PathPrefix = "/internal/"
```

ServerPathPrefix(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 ServerPathPrefix 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다. ReverseProxyGroupName 설정을 사용하는 경우 ServerPathPrefix 설정은 REVERSE_PROXY_GROUP 절에 설정한 값과 동일해야 한다.
- HTTP Request URL의 PathPrefix에 해당하는 부분이 ServerPathPrefix에 설정된 값으로 교체된 후 수정된 요청이 내부 서버(ServerAddress)로 전달된다.
- 다음의 설정으로 HTTP Request URL이 "/internal/abc.html"이 "/docs/abc.html"로 변경된 후 요청이 내부 서버로 전송된다.

```
ServerPathPrefix = "/docs/"
```

ServerAddress(필수 항목)

- 종류: Literal
- 범위: 255자 이내
- 요청이 전달되는 내부 서버 주소를 설정한다.

```
ServerAddress = "internal.server.com:80"
```

ReverseProxyGroupName

- 종류: Literal
- 범위: 31자 이내
- REVERSE_PROXY_GROUP 절에 정의된 이름을 설정한다.
- 여러 Reverse Proxy를 Group으로 묶어서 멀티 서버를 구성하고자 하는 경우 사용한다.

```
ReverseProxyGroupName = "rproxyGroup1"
```

RegExp

- 종류: Literal
- 범위: 511자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 RegExp 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다. ReverseProxyGroupName 설정을 사용하는 경우 RegExp 설정은 REVERSE_PROXY_GROUP 절에 설정한 값과 동일해야 한다.
- HTTP Request URL이 Regular expression과 매칭되면 해당 요청을 Reverse Proxy로 처리한다.
- '!'로 시작하는 경우 exclude를 의미한다.

```
RegExp = "\.(do|jsp)$"
```

SetHostHeader

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하고, SetHostHeader 설정을 하지 않은 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- Reverse Proxy를 사용하여 내부 서버로 요청을 포워딩할 때 Host Header의 값을 설정한다. 설정하지 않은 경우 Host Header의 값은 ServerAddress에 설정한 값이 설정된다.

  "$BypassHostHeader"로 설정한 경우 클라이언트가 보낸 Host Header의 값을 그대로 사용한다.

```
SetHostHeader = "reverseproxy.server.com"
```

VhostName

- 종류: String
- 범위: 1023자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 VhostName 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다. ReverseProxyGroupName 설정을 사용하는 경우 VhostName 설정은 REVERSE_PROXY_GROUP 절에 설정한 값과 동일해야 한다.
- Reverse Proxy가 속하는 VHOST 절에 설정된 Virtual Host의 이름을 설정한다.
- 콤마(,)로 구분하여 64개까지 동시에 설정할 수 있다.

HttpInBufSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 16384
- 사용자의 Request를 받을 때 사용하는 버퍼의 크기를 설정한다.
- 0으로 설정하거나 16777216(16MB)보다 크게 설정할 경우 16777216(16MB)를 사용한다.

RewriteRedirect

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 RewriteRedirect 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- 내부 서버 응답의 Status Code가 301, 302, 303, 307일 경우 (redirect), Location, Content-Location header field의 absolute URL을 수정한다.
- RewriteRedirect는 흰 공백(whitespace)으로 2개의 string으로 나누어진다.

  첫 번째는 원본 응답의 Location, Content-Location의 URL과 비교된다. URL이 정해진 string으로 시작하면 URL이 수정된다. URL Host 부분은 요청에 사용된 Host, Port로 전환되고, URL Path 부분은 두 번째 string으로 교체된다.

- 다음의 설정으로 원본 응답의 Location인 "http://internal.server.com:80/docs/abc.html"이 "http://webtob:8100/internal/abc.html"로 변경된다.

```
RewriteRedirect = "http://internal.server.com:80/docs/ /internal/"
RewriteRedirect = "http://internal.server.com:80/docs_kr/ /internal_kr/"
RewriteRedirect = "http://internal.server.com:80/docs_ch/ /internal_ch/"
```

- RewriteRedirect 설정은 최대 16개까지 설정할 수 있다.

RewriteCookieDomain

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 RewriteCookieDomain 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- 내부 서버 응답의 Cookie Header 필드 값의 "domain=" 값을 수정한다. "domain="에 지정된 domain string이 RewriteCookieDomain에 지정된 string과 일치할 경우 "domain=" 값을 사용자 요청의 domain으로 교체한다.
- 다음과 같이 설정하면 WebtoB의 domain이 webtob인 경우 원본 응답의 Cookie인 "jsessionid=abc, domain=internal.server.com"는 "jsessionid=abc, domain=webtob"로 변경된다.

```
RewriteCookieDomain = "internal.server.com"
```

RewriteCookiePath

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 RewriteCookiePath 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- 내부 서버 응답의 Cookie Header 필드 값의 "path=" 값을 수정한다. "path="에 지정된 string이 RewriteCookiePath 에 지정된 첫 번째 string으로 시작할 경우 두 번째 string으로 교체한다.
- 다음과 같이 설정하면 원본 응답의 Cookie의 "jsessionid=abc, path=/jeus/application"이 "jsessionid=abc, path=/jeus_proxy/application"으로 변경된다.

```
RewriteCookiePath = "/jeus /jeus_proxy"
```

RewriteHtmlMaxSize

- 종류: Numeric
- 단위: bytes
- 범위: 1 ~ INT_MAX

- 기본값: 10240
- ReverseProxyGroupName 설정을 사용하는 경우 RewriteHtmlMaxSize 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- 응답이 HTML 페이지인 경우 (Content-Type: text/html), 페이지 내부의 특정 태그 값들을 수정할 수 있는데, 이때 사용되는 내부 버퍼의 최대 크기를 설정한다.
- 응답이 설정된 값보다 크면, 수정되지 않은 원본 응답이 클라이언트로 전송된다.

RewriteHtmlUrl

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 RewriteHtmlUrl 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- HTML 페이지에 포함된 URL들을 변경할 때 사용된다.
- URL이 설정된 첫 번째 string으로 시작하면, 해당하는 부분이 두 번째 string으로 교체된다. URL이 Host를 포함할 경우 Host는 요청에 사용된 WebtoB 서버 주소로 교체된다.
- URL을 포함하는 tag, attribute은 HtmlUrl을 사용하여 정의한다.
- 다음 Meta tag의 content는 예외로 항상 URL을 포함한다고 가정한다.

```
<meta ... http-equiv="refresh" ... content="http://internal.server.com:80/docs/link3.html" ...>
```

```
RewriteHtmlUrl = "http://test2:80/ /proxy/",
RewriteHtmlUrl = "/ /proxy/"
```

- RewriteRedirect 설정은 최대 32개까지 설정할 수 있다.

HtmlUrl

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하는 경우 HtmlUrl 설정을 하지 않아도 되며 이 경우 REVERSE_PROXY_GROUP 절에 설정한 값을 사용한다.
- HTML 문서 내부의 URL을 포함하는 tag, attribute 이름을 정한다.

  첫 번째 string은 tag 이름이고, 두 번째부터는 attribute 이름을 설정한다.

- 하나 이상의 tag를 설정하려면, HtmlUrl을 여러 번 설정하여 최대 64개까지 설정할 수 있다.

```
HtmlUrl = "a href",
```

```
HtmlUrl = "img src longdesc usemap"
```

StickySessionRoutingId

- 종류: Literal
- 범위: 255자 이내
- ReverseProxyGroupName 설정을 사용하여 여러 내부 서버(WAS)를 연동하는 경우 Sticky Session routing을 사용할 경우 설정한다.
- 내부 서버(WAS)에서 Set-Cookie 응답 헤더에 Sticky Session id(JSESSIONID)로 넣어주는 값의 엔진명을 사용한다. 예를 들어 Set-Cookie 응답 헤더의 "JSESSIONID" 값이 "Pl1xfBkEVbUu2cj20CUNlHJoWLmU.xxx_servlet_engine1"이라면 점(dot)으로 구분하여 이후 값인 "xxx_servlet_engine1"을 입력한다.

```
StickySessionRoutingId = "xxx_servlet_engine1"
```

MinPersistentServerConnections

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 요청 처리후 내부 서버와의 커넥션을 지속적으로 유지하고자 하는 경우 최소 개수를 설정한다.
- 신규 요청이 들어오면 내부 서버와의 커넥션이 새로 맺어지고, 내부 서버가 커넥션을 끊지 않는 이상 해당 커넥션을 지속적으로 유지하며 이후 요청에 대하여 이 커넥션을 재사용한다.
- 내부 서버와의 커넥션을 지속적으로 유지하기 위해 PING 메시지를 보내야 하는데 이를 위해 PersistentServerCheckUrl 설정을 반드시 해야 한다.

MaxPersistentServerConnections

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 요청 처리후 내부 서버와의 커넥션을 지속적으로 유지하고자 하는 경우 최대 개수를 설정한다.
- 재사용 가능한 커넥션이 모두 다른 요청을 처리 중이라면 내부 서버와의 커넥션을 새로 맺어서 추가하며, 이미 최대 개수가 재사용 중이라면 요청은 큐잉된다.
- 내부 서버와의 커넥션을 지속적으로 유지하기 위해 PING 메시지를 보내야 하는데 이를 위해 PersistentServerCheckUrl 설정을 반드시 해야 한다.

PersistentServerCheckTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 30
- 내부 서버와의 커넥션을 체크하는 시간을 설정하며 내부 서버와의 커넥션을 확인 및 지속적으로 유지하는 데 사용된다.
- 내부 서버와의 커넥션을 지속적으로 유지하기 위해서는 내부 서버의 KeepAliveTimeout보다 작게 설정한다. Ready 상태의 커넥션에 대하여 한 번의 PING 메시지를 보내고 PersistentServerCheckTime이 지나도록 응답이 오지 않으면 해당 커넥션은 이상이 발생했다고 판단하여 끊는다.
- 0으로 설정한 경우 Ping 메시지를 보내지 않겠다는 의미이다. 이 경우 내부 서버의 KeepAliveTimeout만큼만 커넥션을 지속하게 된다.

PersistentServerTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 300
- 내부 서버와의 커넥션이 MinPersistentServerConnections보다 많을 경우 Ready 상태가 지속되고 있는 커넥션에 대해 이를 끊고자 하는 경우 설정한다.
- 0으로 설정한 경우 idle connection에 대한 TimeOut을 체크하지 않는다.

PersistentServerCheckUrl

- 종류: Literal
- 범위: 255자 이내
- 내부 서버와의 커넥션을 지속적으로 유지하기 위해 내부적으로 HTTP HEAD 요청을 PING 메시지로 사용한다.
- HTTP 요청을 내부 서버로 보내 HTTP 응답을 받아 이를 PONG 메시지로 확인하고 커넥션을 지속적으로 유지한다. 200 응답이 아닌 경우 커넥션은 지속적으로 유지 되지 않는다. 이를 위해 내부 서버에 Ping 요청에 대해 응답할 수 있는 애플리케이션을 만들어놓아야 한다.

```
PersistentServerCheckUrl = "/ping.html"
```

ProxySslFlag

- 종류: Boolean

- 기본값: N
- 내부 서버와 SSL/TLS 프로토콜을 사용하여 연결할지 여부를 설정한다.
- ProxySslName 설정으로 적용할 PROXY_SSL 절 항목을 지정할 수 있다.

ProxySslName

- 종류: String
- 범위: 31자 이내
- ProxySslFlag를 'Y'로 설정하는 경우 적용되며, 사용할 PROXY_SSL 절 항목을 설정한다.

MaxWebSocketConnections

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 0
- HTTP connection에서 웹 소켓 커넥션으로 업그레이드된 경우 해당 커넥션 수에 대한 제한을 설정한다.
- 0으로 설정한 경우 웹 소켓 커넥션 수를 제한하지 않는다.

WebSocketSessionTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 0
- 웹 소켓 커넥션에 대한 TimeOut을 설정한다. **내부 서버의 WebSocket session에 대한 TimeOut 설정과 맞춰서 설정할 것을 권장한다.**
- 0으로 설정한 경우 TimeOut을 체크하지 않는다.

Options

- 종류: Literal
- 범위: 255자 이내
- 해당 서버에 적용된 옵션을 설정한다.

옵션	설명
{+|-}Cache	Content의 캐시 유무(+|-)를 결정한다.

옵션	설명
DynamicServerAddress	내부 서버와 연결에 실패하는 경우 DNS resolution을 다시 시도한다.

ConnectRetryCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 3
- 설정된 내부 서버와 TCP 연결에 실패하는 경우 DNS resolution을 다시 실행 하는 횟수를 지정할 수 있다.

ConnectTimeout

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 5
- 내부 서버와 TCP 연결을 시도한 후 일정시간 동안 접속 요구에 대한 응답이 없으면 연결을 재시도하도록 시간을 설정한다.
- 해당 시간 안에 내부 서버로부터 접속 요구에 대한 응답이 오지 않은 경우 DNS resolution 및 TCP 연결을 재시도한다.

AccessName

- 종류: String
- 범위: 31자 이내
- 특정 IP 주소에서 들어온 Reverse Proxy 요청에 대하여 허용 또는 거부하는 ACCESS 절 이름을 설정한다.

Headers

- 종류: Literal
- 범위: 255자 이내
- 적용할 HEADERS 절 이름을 설정한다.
- 콤마(,)로 구분하여 15개까지 설정할 수 있다.

Compression

- 종류: Literal

- 범위: 255자 이내
- 응답을 압축할 대상을 결정하는 설정이다.
- 압축할 MIME-Type을 설정하며, 응답의 Content-Type이 사용된다. 해당되는 응답은 클라이언트로 전송되기 전에 GZIP을 사용해서 압축된다.
- 콤마(,)로 구분하여 최대 32개까지 여러 MIME-Type을 지정할 수 있다.
- 압축기능을 사용할 경우 네트워크 트래픽을 줄일 수 있지만, 서버의 성능은 저하될 수 있다.

  압축률이 낮은 파일(zip 같은 압축 파일이나 jpeg 같은 압축 이미지 등)을 압축할 경우 서버에 부하만 주기 때문에 해당 MIME-Type이 압축 대상이 되지 않도록 주의해야 한다.

압축 기능은 HTTP Request Header 중 Accept-Encoding에 GZIP이나 deflate로 지정된 요청에 대해서만 적용된다.

CompressionMinSize

- 종류: Numeric
- 범위: 1 ~ INT_MAX
- 기본값: 1
- 압축 설정을 통해 응답을 압축하고자 할 때 응답의 최소 크기를 설정한다.
- Content-Length 응답 헤더의 값이 설정된 값보다 크면 응답을 압축한다. 단, chunked 응답인 경우 응답의 크기를 알기 어려우므로 적용되지 않는다.

3.23.2. 예제

다음은 REVERSE_PROXY 절을 설정한 예제이다.

```
*REVERSE_PROXY
rproxy1
    ReverseProxyGroupName = "rpg_internal",
    VhostName = "vhost1",
    PathPrefix = "/internal/",
    RegExp = "\.(do|jsp)$",
    ServerAddress = "internal.server.com:80",
    ServerPathPrefix = "/docs/",
    RewriteRedirect = "http://internal.server.com:80/docs/ /internal/",
    RewriteCookieDomain = "internal.server.com",
    RewriteCookiePath = "/jeus /jeus_proxy",
    RewriteHtmlUrl = "http://internal.server.com:80/ /internal/",
    RewriteHtmlUrl = "/ /internal/",
    HtmlUrl = "a href",
    HtmlUrl = "img src longdesc usemap",
    RewriteHtmlMaxSize = 4194304,
    MinPersistentServerConnections = 10,
    MaxPersistentServerConnections = 20,
    PersistentServerCheckTime = 50,
    PersistentServerTimeout = 300,
    PersistentServerCheckUrl = "/ping.html",
```

```
    ProxySslFlag = Y,
    ProxySslName = proxy_ssl1,
    MaxWebSocketConnections = 20,
    WebSocketSessionTimeout = 300,
    Options = "-cache"
```

3.24. LOGLEVEL 절

시스템 로그 메시지 출력을 제어하는 로그 레벨을 설정하는 절이다.

3.24.1. 설정 항목

다음은 LOGLEVEL 절의 환경설정 형식이다.

```
*LOGLEVEL
name    # String[256]
    Level = String[256]               # "INFO"
   #Options = String[256],            # PM.LIST
   #RotateBySeconds = Numeric         # 300 (0-)
   #RotateByFileSize = Numeric        # 0 (0-)
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

LOGLEVEL 절 이름

- 종류: String
- WebtoB 사용하는 로거 (logger) 이름을 설정한다.

로거 이름	설명
. | ROOT	모든 WebtoB 로거이다.
.hth | HTH	HTH 관련 부분의 로그를 담당하는 로거이다.
.htl | HTL	HTL 관련 부분의 로그를 담당하는 로거이다.
.wsm | WSM	WSM 관련 부분의 로그를 담당하는 로거이다.
.server | SERVER	모든 서버 로그를 담당하는 로거이다.
.server.cgis | CGIS	CGIS 서버의 로그를 담당하는 로거이다.
.server.phps | PHPS	PHPS 서버의 로그를 담당하는 로거이다.
.server.ssis | SSIS	SSIS 서버의 로그를 담당하는 로거이다.
.server.filters | FILTERS	FILTERS 서버의 로그를 담당하는 로거이다.

Level(필수 항목)

- 종류: String
- 기본값: "INFO"
- 로그 레벨을 설정한다.
- 시스템 로그 메시지의 레벨이 설정된 값 이상이면 출력된다. 레벨이 낮은 순서대로 "TRACE", "DEBUG", "INFO", "WARN", "FATAL"을 지원한다. 레벨이 낮을수록 자세한 로그 메시지들이 출력된다. 모든 로거의 기본 레벨은 "INFO"이다.
- 가장 낮은 로그 레벨 "TRACE"인 경우 메시지는 시스템 로그 파일에 저장되지 않고, 환경변수 **WEBTOB_TRACE**에 설정된 파일(TRACE 파일)에 저장된다.

  WEBTOB_TRACE에 설정되어 있지 않으면 "$WEBTOBDIR/log/trace/ProcessnamePid-dateTtime.trace"에 저장된다.

  WEBTOB_TRACE는 다음의 Format String을 사용할 수 있다.

Format String	설명
%N%	프로세스 이름이다.
%P%	프로세스 ID이다.
%D%	date이다.

  다음과 같이 설정하면, HTH 메시지는 HTH.trace, CGIS 메시지는 CGIS.trace에 저장된다.

```
WEBTOB_TRACE = %N%.trace
```

Options

- 종류: String
- 로거 작동방식에 추가적인 설정을 한다.
- 다음은 ".hth" 로거에 사용되는 옵션에 대한 설명이다.

옵션	설명
dcR	클라이언트가 보낸 데이터를 TRACE 파일에 출력한다.
dcW	클라이언트로 전송된 데이터를 TRACE 파일에 출력한다.
dsR	서버가 보낸 데이터를 TRACE 파일에 출력한다.
dsW	서버로 전송된 데이터를 TRACE 파일에 출력한다.

- 다음은 옵션을 설정한 예이다.

```
Options = "dcR,dcW,dsR,dsW"
```

RotateBySeconds

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 300 (0은 이 기능을 사용하지 않는것을 의미한다.)
- 새로운 로그 메시지 생성할 때 현재 존재하는 로그 파일이 지정된 시간보다 오래되면 새로운 로그 파일을 생성하도록 설정한다.

RotateByFileSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 0 (0은 이 기능을 사용하지 않는것을 의미한다.)
- 새로운 로그 메시지 생성할 때 현재 존재하는 로그 파일의 크기가 지정된 크기보다 큰 경우 새로운 로그 파일을 생성하도록 설정한다.

3.24.2. 예제

다음은 LOGLEVEL 절을 설정한 예제이다.

```
*LOGLEVEL
.hth
    Level = "DEBUG"
.server.cgis
    Level = "INFO"
```

3.25. HEADERS 절

사용자 요청 및 응답에 특정 HTTP Header를 추가하는 경우 HEADERS 절을 정의하여, NODE 절이나 VHOST 절 또는 SERVER 절에 Headers 항목을 설정하면 된다.

3.25.1. 설정 항목

다음은 HEADERS 절의 환경설정 형식이다.

```
*HEADERS
name    # String[256]
    Action = String[256],
   #FieldName = String[256],
```

```
    #FieldValue = String[1024],
    #RegExp = Literal[512],
    #StatusCode = String[256]           # (HTTP status code)
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

HEADERS 절 이름

- 종류: String
- 범위: 255자 이내
- 정의한 HEADERS 절 이름을 설정한다.

Action(필수 항목)

- 종류: String
- 범위: 255자 이내
- 다음은 종류별 Action에 대한 설명이다. Action 종류에 따라 FieldValue는 필수 항목일 수 있다.

Action	설명
AddRequest	요청에 설정된 HTTP 헤더를 추가한다. 설정된 헤더가 요청에 이미 존재하는 경우 같은 헤더가 추가되기 때문에 사용에 유의해야 한다.
AddResponse	응답에 설정된 HTTP 헤더를 추가한다. 설정된 헤더가 응답에 이미 존재하는 경우 같은 헤더가 추가되기 때문에 사용에 유의해야 한다.
AddIfAbsentRequest	요청에 설정된 HTTP 헤더가 없는 경우 헤더를 추가한다.
AddIfAbsentResponse	응답에 설정된 HTTP 헤더가 없는 경우 헤더를 추가한다.
AppendResponse	이미 존재하는 헤더의 값 뒤에 설정된 FieldValue를 붙인다. 해당 헤더가 없을 경우 action을 취하지 않는다.
EchoResponse	요청에 설정된 HTTP 헤더가 존재하는 경우 응답에 같은 헤더 및 값을 추가한다. 해당 Action은 FieldValue를 설정하지 않는다.
SetResponse	응답에 설정된 HTTP 헤더가 없는 경우 헤더를 추가한다. 이미 존재하는 경우 헤더의 값을 설정된 FieldValue로 치환한다.
UnsetResponse	응답에 설정된 HTTP 헤더가 존재하는 경우 해당 헤더를 삭제한다. 해당 Action은 FieldValue를 설정하지 않는다.

FieldName

- 종류: String
- 범위: 255자 이내

- 추가할 HTTP Header의 이름을 설정한다.

FieldValue

- 종류: String
- 범위: 1023자 이내
- 추가할 HTTP Header의 값을 설정한다.
- 문자열 혹은 기호를 값으로 사용한다. 사용 가능한 기호와 기능에 대한 자세한 내용은 "*LOGGING.Format 항목"을 참고한다.

RegExp

- 종류: Literal
- 범위: 511자 이내
- HTTP Request URL이 Regular expression 패턴과 매칭되는 경우 Header를 추가할 경우 설정한다.

StatusCode

- 종류: String
- 범위: 255자 이내
- 특정 HTTP Status Code일 경우 Header를 추가할 경우 설정한다.

3.25.2. 예제

다음은 HEADERS 절을 설정한 예제이다.

```
* HEADERS
header1         Action="AddIfAbsentRequest",
                FieldName="Test_Header",
                FieldValue="test"

header2         Action="AddIfAbsentResponse",
                FieldName="Test_Header",
                FieldValue="test"

header3         Action="AppendResponse",
                FieldName="Test_Header",
                FieldValue=", AppendResponse"

header4         Action="EchoResponse",
                FieldName="Test_Header"

header5         Action = "AddIfAbsentResponse",
                FieldName = "Access-Control-Allow-Origin",
```

```
                FieldValue = "%{HEADER_FIELD}i"
```

3.26. PRECEDING_COMMAND 절

WebtoB를 기동할 때 WebtoB의 각 프로세스별 CPU Affinity를 지정하는 등의 선행 명령어를 실행하도록 설정한다.

3.26.1. 설정 항목

다음은 PRECEDING_COMMAND 절의 환경설정 형식이다.

```
*PRECEDING_COMMAND
name    # String[256]
    Command = Literal[256]            # MULTI[100]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

PRECEDING_COMMAND 절 이름

- 종류: String
- WebtoB 프로세스(process) 이름을 설정한다.

프로세스 이름	설명
.hth | HTH	HTH 프로세스이다.
.htl | HTL	HTL 프로세스이다.
.wsm | WSM	WSM 프로세스이다.
.server | SERVER	서버 프로세스이다.

Command

- 종류: Literal
- 범위: 255자 이내
- 프로세스 실행할 때 앞서 수행하기 위한 명령어를 입력한다.
- 입력할 명령어는 OS마다 다르다. 명령어는 OS에서 위치한 전체 경로를 입력해야 한다.

OS	명령어
Linux	taskset
AIX	execrset

OS	명령어
HP-UX	psrset
SunOS	psrset

- HTH의 경우 명령어를 여러 번 설정할 수 있으며 최대 100개까지 설정할 수 있다. HTH 이외의 프로세스는 명령어를 여러 번 설정하더라도 가장 첫 번째 명령어만 적용된다.
- NODE 절에 HTH 를 2개 이상 설정한 경우 HTH를 설정한 수보다 명령어 수가 더 많다면 HTH 수만큼의 명령어만 적용된다. 그렇지 않고 HTH 설정한 수가 명령어 수보다 많다면 설정된 명령어를 위에서부터 돌아가면서 순차적으로(마치 Rournd Robin 방식처럼) 적용된다.

  '.hth'에 다음과 같이 2개의 명령어를 설정하고, NODE 절에 HTH를 5로 설정한 경우 hth1/hth3/hth5는 첫 번째 명령어가 적용되고, hth2/hth4는 두 번째 명령어가 적용된다 .

```
Command = "/usr/bin/execrset -c 0 -e",
Command = "/usr/bin/execrset -c 1 -e"
```

- 프로세스가 비정상 종료 후 재기동되는 경우에도 해당 명령어가 동일하게 적용된다.

3.26.2. 예제

다음은 PRECEDING_COMMAND 절을 설정한 예제이다.

```
*PRECEDING_COMMAND
.wsm
    Command = "/usr/bin/taskset -c 1"
.htl
    Command = "/usr/bin/taskset -c 1"
.hth
    Command = "/usr/bin/taskset -c 2",
    Command = "/usr/bin/taskset -c 3",
    Command = "/usr/bin/taskset -c 4"
```

3.27. FILTER 절

Filter 모듈을 사용할 FILTER 절을 정의하여, NODE 절이나 VHOST 절 또는 SVRGROUP 절에 Filter 항목을 설정한다.

3.27.1. 설정 항목

다음은 FILTER 절의 환경설정 형식이다.

```
*FILTER
name    # String[256]
    RealPath = Literal[256]             # $ENV, R.PATH
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

FILTER 절 이름

- 종류: String
- 범위: 255자 이내
- 정의한 FILTER 절 이름을 설정한다.

RealPath(필수 항목)

- 종류: String
- 범위: 255자 이내
- 서버 안의 물리적 디렉터리의 Filter 모듈이 위치한 경로명을 설정한다.

3.27.2. 예제

다음은 FILTER 절을 설정한 예제이다.

```
*FILTER
sm_filter
    RealPath = "/usr/local/webtob/config/filter/wbSmISAPI.so"
```

3.28. USERLOGFORMAT 절

공통으로 사용될 로그 포맷을 설정할 수 있도록 하며 절에서 정의한 포맷은 LOGGING 절에서 사용할 수 있다.

3.28.1. 설정 항목

```
*USERLOGFORMAT
name    # String[256]
    Format = Literal[256]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

USERLOGFORMAT 절 이름

- 종류 : String

- 범위 : 32자 이내
- 정의할 로그 포맷의 이름을 설정한다.
- 설정한 이름을 *LOGGING.Format의 인자로 사용할 수 있다.

Format(필수 항목)

- 종류 : String
- 범위 : 255자 이내
- 로그 메시지의 포맷을 설정한다. 자세한 메시지의 내용은 "*LOGGING.Format 항목"을 참고한다.

3.28.2. 예제

다음은 USERLOGFORMAT 절을 설정한 예제이다.

```
*USERLOFGFORMAT
format1
    Format = "%h %l %u %t \"%r\" %s %b"
```

3.29. DISK_CACHE 절

WebtoB에서 디스크 캐시 기능을 사용할 경우 설정한다. 디스크 캐시는 파일이 NAS, NFS 등과 같이 로컬이 아닌 경로(네트워크 이용)에 있을 경우 로컬 디스크에 캐시하고자 하는 경우 사용한다. 이 설정을 통해 캐시에 사용 될 디스크 경로, 캐시할 수 있는 파일 크기 등을 설정할 수 있다.

3.29.1. 설정 항목

```
*DISK_CACHE
name    # String[32]
    CacheRoot = Literal[256]            # "cache_root/", $ENV, R.PATH
    DirLevels = Numeric                 # 0 (0 - )
    MinFileSize = Numeric               # 1 (1 - )
    MaxFileSize = Numeric               # 1000000
    RefreshTime = Numeric               # 1 (1 - )
    RetryWaitCount = Numeric            # 5
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

DISK_CACHE 절 이름

- 종류 : String

- 범위 : 32자 이내
- 사용할 디스크 캐시의 이름을 설정한다.

CacheRoot (필수 항목)

- 종류 : String
- 범위 : 255자 이내
- 디스크 캐시에 사용할 경로를 설정한다.
- 환경변수를 사용할 수 있으며, 상대 경로로 설정한 경우 $WEBTOBDIR을 기준으로 절대 경로로 변환하여 설정된다.

DirLevels

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값 : 0
- 디스크 캐시를 할 때 한 번에 만들 수 있는 디렉터리의 depth를 설정한다.

MinFileSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ INT_MAX
- 기본값: 1
- 디스크 캐시할 수 있는 파일의 최소 크기를 설정한다.

MaxFileSize

- 종류: Numeric
- 단위: bytes
- 범위: 0 ~ LONG_MAX
- 기본값: 1000000
- 디스크 캐시할 수 있는 파일의 최대 크기를 설정한다.

RefreshTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 1
- 디스크 캐시된 응답에 대한 유효시간을 설정한다. 즉, 캐시된 후 설정한 시간(초) 동안만 유효하다.
- 0으로 설정하는 경우 디스크 캐시된 응답은 만료시간 없이 유효하다고 판단한다.

RetryWaitCount

- 종류: Numeric
- 범위: 0 ~ INT_MAX
- 기본값: 5
- 서로 다른 클라이언트로부터 같은 파일을 요청할 경우 먼저 들어온 요청에 의해 디스크 캐시가 이루어진다.
- 이럴 경우 이후의 요청은 최초 요청이 디스크 캐시를 하고 응답을 할 때까지 대기-재요청을 반복하게 되고 이러한 대기-재요청의 최대 횟수를 설정한다,

3.29.2. 예제

다음은 DISK_CACHE 절을 설정한 예제이다.

```
*DISK_CACHE
dc1
    CacheRoot = "cache/node",
    DirLevels = 5,
    MinFileSize = 8193,
    MaxFileSize = 100000000,
    RefreshTime = 3600
```

3.30. LOG_HANDLER 절

WebtoB accesslog를 사용자가 지정한 리모트 서버에 남기고자 하는 경우 설정한다. 이 설정을 통해 리모트 서버의 주소와 로그를 남길 경로, connection timeout 등을 설정할 수 있다.

3.30.1. 설정 항목

```
*LOG_HANDLER
name    # String[32]
   ServerAddress = Literal[256],
   #ConnectTimeout = Numeric,      # 3 (0-)
   #ReconnectTime = Numeric,       # 10
   #Timeout = Numeric,             # 60
   Resource = Literal[64],
```

```
Protocol = Literal[64]        # HTTP
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

LOG_HANDLER 절 이름

- 종류 : Literal
- 범위 : 32자 이내
- 사용할 LOG_HANDLER 이름을 설정한다.

ServerAddress (필수 항목)

- 종류 : Literal
- 범위 : 255자 이내
- accesslog를 남길 리모트 서버의 주소를 설정한다.

```
ServerAddress = "remote.server.com:80"
```

ConnectTimeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값 : 3
- 리모트 서버로 연결 요청 후 연결될 때까지 기다리는 시간을 설정한다.

ReconnectTime

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 10
- 리모트 서버와 연결 실패시 다시 연결을 요청할 때까지 기다리는 시간을 설정한다.

Timeout

- 종류: Numeric
- 단위: 초
- 범위: 0 ~ INT_MAX
- 기본값: 60
- 설정한 시간이 지나도록 accesslog 기록 요청이 발생하지 않으면 리모트 서버와 연결을 끊는다.

Resource (필수 항목)

- 종류: Literal
- 범위: 63자 이내
- 리모트 서버에서 accesslog를 저장할 경로를 설정한다.

Protocol (필수 항목)

- 종류: Literal
- 기본값: HTTP
- 리모트 서버에 accesslog 요청을 보낼 때 사용할 프로토콜을 설정한다.

3.30.2. 예제

다음은 LOG_HANDLER 절을 설정한 예제이다.

```
*LOG_HANDLER
lh1
    ServerAddress = "remote.server.com:8080",
    ConnectTimeout = 3,
    ReconnectTime = 10,
    Timeout = 60,
    Resource = "webtob/accesslog",
    Protocol = "HTTP"
```

3.31. IPGROUP 절

ACCESS 절에 설정하는 IP가 많은 경우 필요에 따라 이를 그룹화하여 설정할 수 있다.

3.31.1. 설정 항목

```
*IPGROUP
name    # String[32]
```

```
        IP = Literal[8960]
```

위의 절과 설정 항목의 구성에 대한 기호나 내용에 대한 자세한 내용은 설정 항목 설명 규약을 참고한다.

IPGROUP 절 이름

- 종류 : String
- 범위 : 32자 이내
- 정의할 IP 그룹의 이름을 설정한다.
- 설정한 이름을 *ACCESS.Allow, *ACCESS.Deny의 인자로 사용할 수 있다.

IP(필수 항목)

- 종류 : String
- 범위 : 8960자 이내
- 그룹화할 IP의 목록을 설정한다. 각 IP는 콤마(,)로 구분된다.

3.31.2. 예제

다음은 IPGROUP 절을 설정한 예제이다.

```
*IPGROUP
ipgroup1
        IP = "localhost, 123.123.123.123"

*ACCESS
access1
        Allow = "100.100.100.100, ipgroup1"
```

4. 고급 설정

본 장에서는 앞에서 살펴본 기본적인 설정 외에 특수한 작업을 하기 위한 설정에 대하여 설명한다.

4.1. 동적 콘텐츠를 위한 설정

웹 서버가 설치되면 HTML을 이용한 서비스가 가능하다. 즉, WebtoB의 Document Root Directory에 HTML 문서를 두면 브라우저로 호출하여 열람이 가능하다. 본 절에서는 CGI, SSI, PHP를 사용하기 위한 설정과 간단한 예제를 통해 그 설치 예에 대해 설명한다.

4.1.1. CGI

CGI(Common Gateway Interface)는 웹에서 HTML은 의해 여러 가지 정보를 표현하고 홈 페이지를 만드는 기초가 되는 언어이다. 하지만 HTML만으로 다양한 정보를 처리할 수 없고 서버로부터 클라이언트에게 단방향의 정보만 제공한다. 이러한 단점을 보완하기 위해 외부 프로그램을 수행하여 그것의 결과를 HTML 형태로 보여주는 방식이 도입되었다. CGI는 외부 프로그램과 웹 서버 간의 연결 역할을 한다.

넓은 의미로 CGI는 수행하는 외부 프로그램을 포함하여 말하기도 한다. 예를 들어, 홈 페이지에 방문객들의 comment를 받을 수 있는 방명록을 만들 때 웹에서 구현하는 HTML만으로는 해결할 수 없다. 그래서 외부 프로그램이 필요한데, 이때 외부 프로그램과 웹 서버 간에 서로 주고받을 수 있는 규약을 CGI라고 하고, 사용하는 프로그램을 흔히 CGI 프로그램(혹은 CGI 스크립트)이라고 한다. CGI 프로그램은 통상적으로 C/C++이나 Perl 혹은 UNIX 쉘, Tcl/Tk 등을 사용하여 구현한다.

홈 페이지를 interative한 형태로 만들 수 있는 CGI 프로그램의 종류는 매우 다양하고 표준화되어 있다. 방문객 카운터나 방명록뿐만 아니라 웹 게시판, 웹 대화방, 검색엔진, 다양한 배너 보여주기, 업로드가 가능한 자료실, 폼을 이용하여 메일을 띄우는 폼 메일(Form Mail) 등이 CGI 프로그램의 예이다.

CGI는 외부 프로그램이 수행되는 방식이기 때문에 프로그램 자체에 문제가 없다면, 다른 곳에 서로 이식할 수 있다. 어떤 웹 서버에서 동작하는 CGI라면 다른 웹 서버도 문제없이 동작한다. WebtoB 역시 다른 웹 서버와 같은 방식으로 처리하기 때문에 CGI 수행에는 전혀 지장이 없다.

CGI를 제작하는 방법은 매우 다양하고, 여러 가지 방법이 있기때문에 본 안내서에서 깊은 내용을 언급하지는 않고 WebtoB에서 CGI를 사용하기 위한 설정 방법에 대해서만 설명한다.

CGI 설정

CGI 사용을 위해 다음의 각 절에 해당 내용을 설정해야 한다.

- **SVRGROUP 절**

  SVRGROUP 절에 CGI 사용을 위한 내용을 설정한다. 설정된 내용은 SERVER 절에서 서버 그룹을 설정하는 데 사용된다.

  다음은 SVRGROUP 절 설정에 대한 예제이다. SvrType 항목을 CGI로 설정하면 서버 그룹은 CGI를 처리한다.

  ```
  *SVRGROUP
  ```

```
cgig        NodeName = mynode,
            SvrType = CGI
```

- **SERVER 절**

  SERVER 절에 서버 프로세스를 설정한다. 다음은 SERVER 절 설정에 대한 예제이다.

```
*SERVER
cgi         SvgName = cgig,
            MinProc = 10,
            MaxProc = 10
```

  cgi라는 이름의 서버 프로세스가 기동된다. SvgName을 SVRGROUP 절에 설정한 cgig로 설정해서 cgi라 명명된 현재의 서버 프로세스는 cgig라는 서버 그룹에 종속된다. cgig가 SvrType이 CGI로 설정되어 있기 때문에 cgi 서버는 CGI 서비스를 한다.

- **URI 절**

  URI 절의 설정을 통해 특정 URI의 요청을 CGI로 처리하도록 설정한다. 다음은 URI가 "/cgi-bin/"로 시작할 경우 CGI로 간주하여 처리하도록 설정한 예제이다.

```
*URI
uri1        Uri = "/cgi-bin/",
            SvrType = CGI
```

- **ALIAS 절**

  ALIAS 절에 특정 URI에 대해서 실제로 처리하게 되는 물리적인 경로를 설정한다. 해당되는 경로가 DocRoot가 아닌 다른 위치에 있을 경우에만 설정한다.

  다음은 ALIAS 절 설정에 대한 예제이다.

```
*ALIAS
alias1      Uri = "/cgi-bin/",
            Realpath = "${WEBTOBDIR}/cgi-bin/"
```

설정은 환경설정 파일 예제, 사용 예는 CGI 활용 예제를 참조한다.

4.1.2. SSI

본 절에서는 SSI의 이용 방법과 설정 방법에 대해서 설명한다.

SSI의 이용

SSI(Server Side Includes)은 HTML 페이지에서 다이내믹한 문서를 만들때 유용하게 사용할 수 있다. 예를 들면 Header 파일 등을 더할 수도 있고, 파일의 마지막 수정시간(Last Modified Tiem)을 자동으로 조절할 수 있는

기능들을 문서에 포함시킬 수도 있다.

SSI는 HTML 문서에 'command'를 집어 넣어 사용할 수 있으며, 서버에서는 SSI 문서를 읽어 들이고, SSI 명령어를 찾아보고 그에 맞는 기능을 수행한다. 예를 들면 파일의 Last Modification Time을 수정하는 SSI 명령어가 문서에 포함되어 있으면, 서버는 파일로부터 명령어들을 읽어 들여 명령을 수행하여 파일의 마지막 수정시간을 갱신한다.

WebtoB에서 기본 설정은 HTML 파일 안에 SSI 명령어를 포함하고 있지 않다. 왜냐하면, HTML 파일을 매일 액세스하는 곳에서는 이로 인하여 HTML이 오히려 느리게 작동하기 때문이다. SSI가 필요해서 사용하기 원한다면 WebtoB 환경 파일에 SSI를 위한 서버를 추가로 설정해야 한다.

다음은 SSI를 사용하기 위한 환경설정 예이다.

```
*SVRGROUP
ssig         NodeName = mynode, SvrType = SSI

*SERVER
ssi          SvgName = ssig, MinProc = 10, MaxProc = 10
```

SSI Commands

모든 SSI 명령어는 HTML 문서 안에 HTML comments 형식으로 저장되어야 한다.

모든 명령어는 다음 형식을 따르며, 명령어 전체는 comment "<!--... -->"로 처리되어야 한다. arg1, arg2는 인수고 나머지 value1, value2는 인수의 값을 나타낸다.

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

다음은 SSI 사용 방법이다. flastmod 명령어는 수정시간을 출력하라는 뜻이고 Value로는 nextel.html이 사용된다. 'file'은 인수가 되는 것이고 'nextel.html'은 값이 된다.

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

다음은 명령어는 인수 이름에 따라 다르게 실행되는 예이다.

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

echo 명령어로 서버 홈 페이지의 마지막 수정시간을 얻을 수 있다. SSI 명령어가 실행될 때 'environment variables' 값이 설정된다. CGI variables을 포함하고 있고(REMOTE_HOST etc), DOCUMENT_NAME 그리고 LAST_MODIFIED 등을 가지고 있다.

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

4.1.3. PHP

본 절에서는 PHP의 이용 방법과 설정 방법에 대해서 설명한다.

PHP의 이용

PHP는 Perl과 유사한 형태의 스크립트 언어로 간편성과 성능으로 인하여 많이 이용되고 있다. 속도, 개발 편의성, 여러 가지 확장 기능 면에서 기존의 Perl보다 발전된 언어로 Linux나 UNIX 계열뿐만 아니라, WIN32용 바이너리 파일을 제공해 Microsoft 계열의 웹 서버에서도 사용된다. PHP는 운영체제에 독립적인 웹 프로그램 개발이 가능한 것이 큰 장점이다.

PHP는 CGI 프로그램에서 할 수 있는 form data를 가져오고 동적인 웹 페이지를 만들거나, Cookie를 보내고 받는 모든 기능의 구현이 가능하다.

PHP의 설정

WebtoB에서 PHP를 이용하기 위해서는 설치 작업이 필요하다. HTML이나 CGI를 이용하는 것과 유사하기 때문에 쉽게 적용할 수 있다.

- **SVRGROUP 절**

  PHP에 관련된 그룹을 설정하고 이를 SEVER 절에서 다시 정의한다.

  SVRGROUP 절에 ScriptLoc 항목이 추가되는데, 이는 PHP의 실제 실행 모듈이 있는 위치이다. PHP 모듈이 있는 곳의 경로는 ${WEBTOBDIR}로부터의 상대 경로를 설정해야 한다. ScriptArgs 항목을 설정하여 PHP의 실행 모듈듈을 실행할 때 적용할 옵션을 지정할 수 있다. php.ini 파일등을 지정할 때는 절대 경로를 사용해야 한다.

  다음 예제에서 ScriptLoc에 설정된 모듈 경로는 절대 경로가 아니라 "${WEBTOBDIR}/cgi-bin/php"를 의미이며, ScriptArgs에 설정된 경로는 실제(절대)경로를 의미한다.

```
*SVRGROUP
phpg            NodeName = mynode,
                ScriptLoc = "/cgi-bin/php", # "${WEBTOBDIR}/cgi-bin/php"를 의미
                ScriptArgs = "-c /php_conf/php.ini", # php 실행시 실행모듈에 대한 옵션
                SvrType = PHP

*SERVER
php             SvgName = phpg, MinProc = 10, MaxProc = 10
```

php3 모듈을 사용하는 경우에는 부가적인 설정이 필요하다. php4는 기본적으로 php라는 확장자를 이용하기 때문에 추가 설정이 필요하지 않으나, php3는 확장자가 ".php3"이기 때문에 파일을 사용하기 위해서 EXT 절에 추가해 주어야 동작한다. 다음은 EXT 절 설정에 대한 예제이다.

```
*EXT
php3            Mimetype = "application/x-httpd-php3",
                SvrType = PHP
```

PHP 예

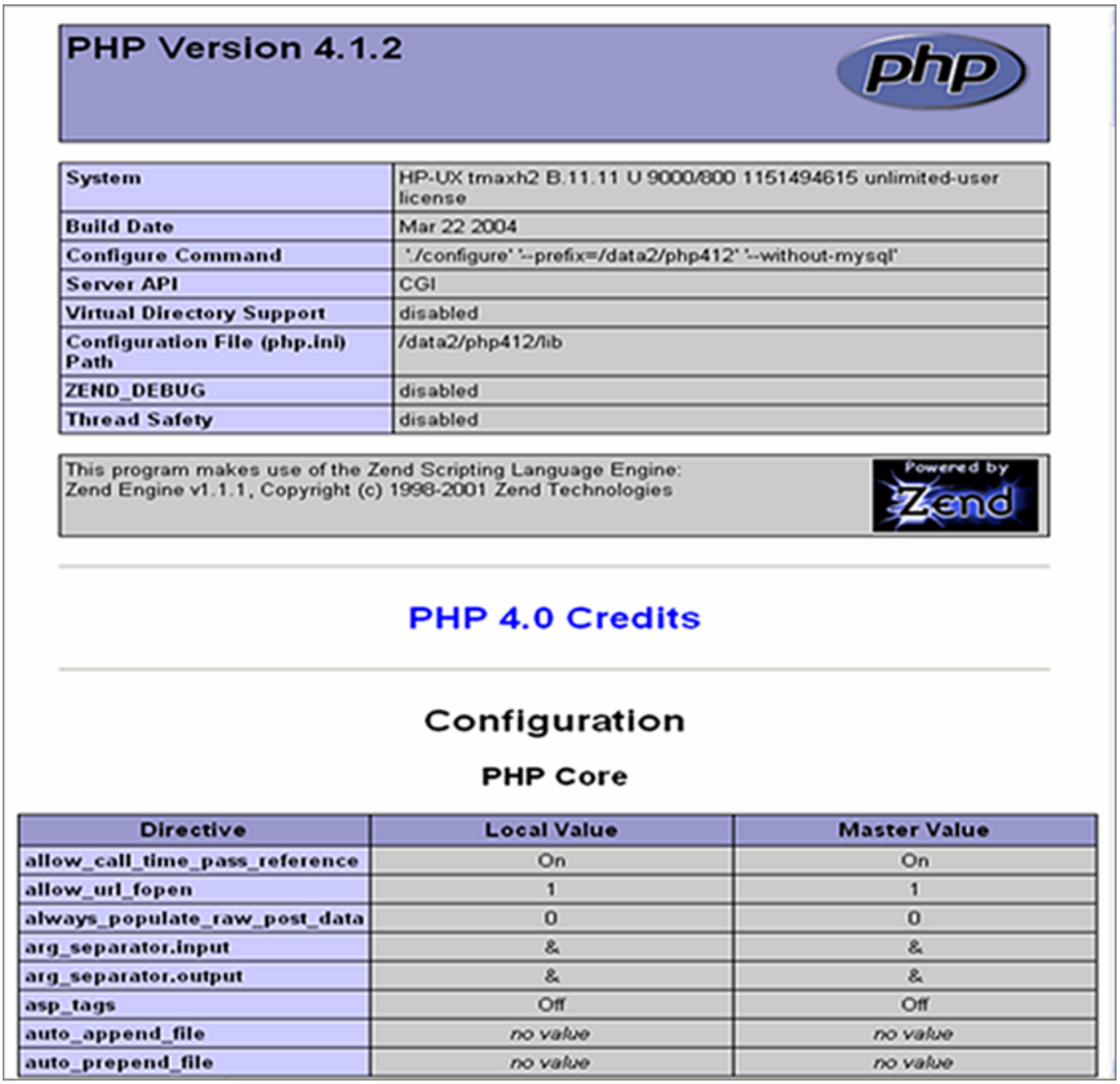
다음의 순서로 PHP의 설치 및 WebtoB에서의 설정이 정상적인지 확인한다.

1. ${WEBTOBDIR}/docs에 다음과 같이 "phpinfo.php"를 만든다.

```
<?
  Phpinfo();
?>
```

2. 브라우저에서 다음과 같이 요청한다. 해당 페이지는 설치된 PHP에 대한 각종 환경정보를 나타낸다.

```
http://<IP address>:<PORT>/phpinfo.php
```



PHP Version 4.1.2

System	HP-UX tmaxh2 B.11.11 U 9000/800 1151494615 unlimited-user license
Build Date	Mar 22 2004
Configure Command	'./configure' '--prefix=/data2/php412' '--without-mysql'
Server API	CGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/data2/php412/lib
ZEND_DEBUG	disabled
Thread Safety	disabled

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.1.1, Copyright (c) 1998-2001 Zend Technologies

PHP 4.0 Credits

Configuration

PHP Core

Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
allow_url_fopen	1	1
always_populate_raw_post_data	0	0
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	*no value*	*no value*
auto_prepend_file	*no value*	*no value*

그림 2. PHP 테스트 페이지

4.2. Virtual Hosting

본 절에서는 Virtual Host의 개념과 사용 방법에 대해서 설명한다.

4.2.1. Virtual Host 구조

가상 호스트(Virtual Hosting)는 현재 HTTP 1.1을 지원하는 브라우저에서 적용할 수 있는 웹 서버의 기능으로 하나의 웹 서버를 이용하여 마치 여러 대의 웹 서버가 운영되고 있는 것과 동일한 효과를 낼 수 있다.

예를 들어, "WebtoB Times"라는 이름으로 신문사 홈 페이지를 운영 중이고, 메인 홈 페이지와 사회면 기사를 다루는 홈 페이지, 스포츠 기사를 다루는 홈 페이지를 도메인 이름으로 구분하여 서비스한다고 가정한다. 이 서비스를 하나의 IP 주소와 도메인 이름, 그리고 1대의 웹 서버를 이용해서 운영하려고 할 경우 Virtual Host 기능을 사용한다.

다음 같이 도메인 이름을 분리하여 HTML 문서도 만들고 기타 서비스도 제공한다.

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

웹 서버에서는 위와 같은 설정을 적용하기 위해 2개의 Virtual Host를 할당하여 각각 society와 sports에 적용해 주면 된다. 이렇게 하면 이들 서비스는 메인 페이지와는 다른 웹 서버에서 운영되는 것과 같은 효과를 줄 수 있으며, 실제 웹 서버 내에서도 문서의 경로 및 기타 모든 설정들을 분리하여 사용할 수 있다. 이렇게 도메인 이름을 분리하여 하나의 IP 주소로 서비스하는 방식을 **Name Based Virtual Host**라고 한다.

이와 구별되는 방식으로 **IP Address Based Virtual Host**가 있는데, 이 방식은 다른 IP 주소를 사용하므로 사실상 별개의 도메인이라 할 수 있다. 하나의 사이트가 다량의 IP 주소를 확보하기는 힘들어 크게 사용되지 않는 방식이다.

다음은 위 의 예에서 구상한 Virtual Host의 구조이다. 하나의 IP와 1대의 웹 서버를 이용해 각 도메인의 서비스를 할 수 있으며, 동일 IP에 다른 이름으로 호출된 서비스를 WebtoB의 Virtual Host가 구분하여 서비스할 수 있다.

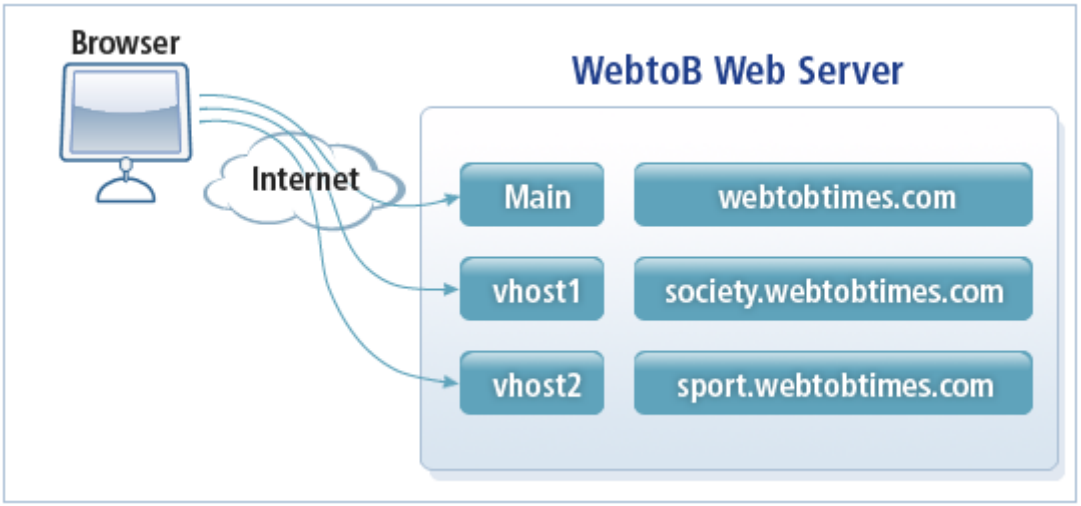


그림 3. Virtual Host의 구조

다음은 위의 "WebtoB Times"는 가상 홈 페이지를 WebtoB에서 구축한 Virtual Host의 설정 예제로 VHOST 절을 적절히 이용해 주면 효과적인 웹 운영이 가능하다.

```
*VHOST
society
    Docroot = "society_docs/",
    NodeName = webtob,
    Hostname = "society.webtobtimes.com",
    Port = "8080",
    IconDir = "society_icons/",
    LOGGING = "society_accesslog",
    ERRORLOG = "society_errorlog"

sports
    Docroot = "sports_docs/",
    NodeName = webtob,
    Hostname = "sports.webtobtimes.com",
    Port = "8080",
    IconDir = "sports_icons",
    LOGGING = "sports_accesslog",
    ERRORLOG = "sports_errorlog"
```

4.2.2. Mass Virtual Host

대량의 Virtual Host를 설정함에 있어 각각의 설정이 비슷한 경우 일일이 설정하지 않고 보다 간편하게 이용할 수 있는 설정 방법이 있다.

- **DocRoot 항목 설정**

  대량의 Virtual Host에 있어서 DocRoot 설정만 다른 경우 이용할 수 있는 기능으로 각 Virtual Host의 DocRoot는 해당 서버의 도메인 이름 값을 디렉터리 설정 패턴으로 치환함으로서 결정한다.

  ```
  Docroot = <literal>    (Default : "docs/")
  ```

  예를 들어 클라이언트에서 http://www.tmax.co.kr로 요청했을 때 이 Virtual Host의 DocRoot를 "${WEBTOBDIR}/docs/www/"로 사용하고 싶을 경우 다음과 같이 설정하면 된다. 마찬가지로 http://webtob.tmax.co.kr로 접속하였다면 DocRoot는 "${WEBTOBDIR}/docs/webtob/"가 된다.

  ```
  *VHOST
  TmaxSoft
      NodeName = tmax,
      Hostname = "www.tmax.co.kr"
      HostAlias = "webtob.tmax.co.kr"
      Port = "80",
      Docroot = "${WEBTOBDIR}/docs/%1/"
  ```

- **Realpath 항목 설정**

  여러 개의 Virtual Host를 사용할 때 ALIAS 절의 Realpath를 설정한다. 서버의 도메인 이름 값을 디렉터리 설정 패턴으로 치환함으로써 RealPath 경로를 설정할 수 있다.

  ```
  Realpath = <literal>    (Default : mandatory)
  ```

DocRoot를 설정했을 경우와 마찬가지로, "%1"이 클라이언트가 요청한 도메인의 첫 번째 항목으로 치환된다.

```
*ALIAS
alias1        Uri = "/cgi-bin",
              Realpath = "${WEBTOBDIR}/docs/%1/cgi-bin/"
```

다음은 디렉터리 설정 패턴에 사용되는 "%" 지시자에 대한 설명이다. 지시자를 사용해서 디렉터리 설정 패턴을 동적으로 설정할 수 있다.

지시자	설명
%p	요청한 포트 번호로 치환한다.
%n	Hostname이나 IP 주소의 n번째 요소로 치환한다. 만약 n을 0으로 하면 전체 문자열이 사용된다. 마이너스 기호(-)가 앞에 오면 Hostname이나 IP 주소의 끝에서부터 센다. 플러스 기호(+)가 뒤에 오면 Hostname이나 IP 주소의 나머지가 사용된다.
%n.m	n번째 요소의 m번째 문자로 치환한다. 위와 같이 마이너스 기호(-)나 플러스 기호(+)가 붙을 수 있다.
%%	단일 퍼센트(%) 표시로 치환한다.

"%" 지시자를 사용하고자 하는 경우 DocRoot/Realpath를 절대 경로로 설정해야 한다.
${env}를 사용할 경우 치환된 경로가 최종적으로 절대 경로면 "%"지시자를 사용할 수 있다.

4.3. 로그 설정

본 절에서는 로그 파일에 대한 개념과 각 포맷별 로그 파일에 대해서 설명한다.

4.3.1. 로그 파일

WebtoB는 웹 서비스에 대한 요청과 응답 등 서비스 관련된 기록을 모두 환경설정에서 지정한 로그 파일에 저장한다. 따라서 로그 파일을 보면 누가 언제 무엇을 요청했고 또 무엇을 가져갔는지 알 수 있으며 웹 서버에 얼마나 많은 접속이 있었는지 가장 많이 보는 페이지는 무엇인지 등을 알 수 있다.

로그 파일은 환경설정에 설정한 위치에 저장되며 웹 서버를 운영할 때 임의의 위치에 저장이 가능하다. 로그를 남기는 과정은 시스템 측면과 Capacity 측면에서 서버에 어느 정도의 부하를 줄 수 있다.

- 시스템 측면에서 로그는 디스크에 기록하는 작업을 해야 하기 때문에 로그를 남기지 않는 것에 비해 속도가 약간 느려지게 된다. WebtoB에선 내부적으로 효율적인 방법을 이용하지만, 한계가 있는 관계로 약간의 속도 저하는 발생할 수 있다.
- Capacity 측면에서는 로그를 남기는 공간에 대한 부하를 고려해야 한다. 접속자가 적은 곳에서는 로그 파일의 크기가 작겠지만, 접속자가 아주 많은 곳에서는 하루에 로그 파일이 몇 백 Mbytes에서 수 Gbyte에 이르는 로그를 남길 수 있다. 시스템에 적당한 공간이 남아있지 않다면 불가능하기 때문에 이에 대한 관리가 중요하다.

로그 파일은 웹 서버 관리자에게는 중요한 정보이다. 운영하는 웹 서버에 접속한 사용자의 정보를 바탕으로 좀 더 효율적인 웹 서비스를 구현할 수도 있고 쇼핑몰 운영자에게는 고객들의 정보를 알 수 있는 아주 귀중한 자료이다. 따라서 WebtoB를 이용하는 운영자들은 로그 파일을 특정 위치에 저장하고 관리한다.

WebtoB는 여러 개의 로그 파일을 생성하는데, 사용자 요청과 관련된 액세스 로그와 요청 처리 중 오류 메시지와

관련된 에러 로그, WebtoB 시스템을 위한 시스템 로그로 나누어진다.

구분	설명
액세스 로그	액세스 로그는 Transfer 로그라고도 한다. 사용자의 요청과 관련된 일반적인 사항을 기록하며, 사용자가 보낸 요청과 관련된 정보와 서버에서 어떤 응답을 보냈는지 등에 관련된 정보를 알 수 있다. 그러므로 이 정보는 차후에도 많은 이용가치가 있기 때문에 아주 중요하다. WebtoB의 NODE, SVRGROUP, VHOST 절 등에 모두 설정이 가능하다. 각 절에 모두 설정했을 경우 접속자의 접속 서비스 요청에 따라 그 우선순위가 SVRGROUP, VHOST, NODE 절 순서로 적용된다. 가장 우선순위가 높은 절에 해당하는 로그 파일에 그 내용이 기록된다.
에러 로그	에러 로그는 사용자 요청을 처리하는 과정에서 발생한 오류에 대해서 기록한다.
시스템 로그	시스템 로그는 WebtoB를 운영 중에 발생한 이벤트를 기록한다. WebtoB를 운영하는 중 문제가 생긴 경우 문제 해결에 큰 도움이 될 수 있는 정보를 제공하게 된다. 따라서 중요한 업무를 수행하는 경우에는 시스템 로그를 저장하여 서버 운영에 문제가 생길 경우 로그를 통하여 문제 해결을 쉽게 할 수 있다. 시스템 로그는 NODE 절에서 설정 가능하다.

4.3.2. Log Format

WebtoB는 LOGGING 절의 Format 항목 설정을 통해 다양한 형태로 액세스 로그를 남길 수 있다.

보통의 경우 Common Log Format(CLF)을 사용하는데, 이는 NCSA 계열의 웹 서버에서 사용하는 Log Format으로 대부분의 웹 서버에서 기본으로 사용하는 Log Format이다. WebtoB에서는 이를 "default"라는 이름으로 제공한다.

4.3.3. Common Log Format

W3C에서 정의하고 있는 Common Log Format은 다음과 같다.

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

다음의 항목 순서로 로그를 생성해서 WebtoB 환경 파일에 지정된 위치에 저장한다.

항목	설명
remotehost	Remote Host 이름을 저장한다(DNS Host 이름을 사용할 수 없거나 DNSLookup이 off이면 IP 주소를 입력한다). Remote Host에 대한 자세한 내용은 http://www.w3.org/Daemon/User/Config/General.html-%20DNSLookup을 참고한다.
rfc931	사용자의 Remote 로그 이름을 저장한다.
authuser	인증된 사용자의 사용자 이름을 저장한다.
[date]	요청한 날짜와 시간을 저장한다.
"request"	클라이언트가 요청한 내용을 저장한다.

항목	설명
status	클라이언트로 응답한 HTTP Status Code를 저장한다. Status Code에 대한 자세한 내용은 http://www.w3.org/Protocols을 참고한다.
bytes	전송한 문서의 콘텐츠 길이를 저장한다.

다음은 WebtoB에서 저장하여 기록하는 로그 파일의 예이다.

```
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 로그 내용에서 맨 앞은 접속한 사용자의 IP 주소를 나타낸 것이고, 뒤의 하이픈(-)들은 각각 rfc931과 AuthUser를 기록해야 한다. 그러나, 사용자에 관련된 정보가 없기 때문에 하이픈(-)으로 기록한다.

뒤에 [ ] 안에 있는 정보가 사용자가 접속한 시간을 나타낸 것이고, 그 뒤에 따옴표("")안의 정보는 사용자 요청 중 HTTP 프로토콜의 Request Line이다. Request Line 뒤의 숫자는 그에 해당하는 Response Status Code이다. 그리고 마지막으로 나타나는 숫자는 사용자에게 서버가 전달한 Byte 수이다.

4.3.4. 환경설정

WebtoB에서 Log Format은 LOGGING 절에 Format 항목에 설정한다.

WebtoB에서 위와 같은 로그를 저장하기 위해서는 다음과 같이 환경설정을 해야 한다.

```
*NODE
webmain
    WebtoBDir = "$WEBTOBDIR",
    SHMKEY = 69000,
    DOCROOT = "docs/",
    Port = "5469",
Logging = "accesslog",
    ErrorLog = "errorlog"

*LOGGING
accesslog
    Format = "default",
    Filename = "log/access.log",
    Option = "Sync"
errorlog
    Format = "",
    Filename = "log/error.log",
    Option = "Sync"
```

NODE 절에 Logging이라는 항목이 바로 액세스 로그를 기록하기 위한 것이다. 이에 대한 설정은 LOGGING 절에서 설정한 "accesslog"로 설정되어 있다. 에러 로그는 ErrorLog라는 항목으로 설정된다. 위의 예에서는 LOGGING 절에서 설정한 "errorlog"로 설정되어 있다.

Format 항목은 다음과 같은 지시자를 사용하여 사용자가 원하는 형태로 액세스 로그를 저장할 수 있다.

포맷	설명
DEFAULT	Default Log File Format이다. (약자: "%h %t \"%r\" %s %b %D")
COMMON	Common Log File Format이다. (약자: "%h %l %u %t \"%r\" %s %b")
COMBINED	Combined Log File Format이다. (약자: "%h %l %u %t \"%r\" %s %b \"%{*Referer*}i\" \"%{*User-Agent*}i\")
%a	요청을 보낸 장비의 IP 주소를 표시한다. %h와 동일하다.
%b	응답의 Byte를 표시한다.
%c	WebtoB 내부 캐시에서 응답이 생성되었는지를 표시한다. 이 경우 "hc"로 표시한다.
%{_attr_name_}C	HTTP Request의 Cookie Header 값 중 '_attr_name_'에 해당하는 값을 표시한다.
%d	응답이 전송된 시간을 표시한다.
%D	요청을 처리하는 데 소요된 시간을 표시한다. (단위: millisecond)
%{ENV_NAME}e	환경변수 ENV_NAME을 출력한다.
%g	WebtoB가 내부적으로 사용하는 요청 식별자를 출력한다.
%h	요청을 보낸 장비의 IP 주소를 표시한다.
%H	사용한 HTTP 버전을 표시한다.
%{HEADER_FIELD}i	HTTP Request의 HEADER_FIELD Header 값을 표시한다.
%{_id_name_}J	요청을 JEUS로 포워딩하여 처리하는 경우 내부적으로 사용하는 요청의 식별정보를 표시한다. ◦ _id_name이 JSVCid이면 Client ID이다. ◦ _id_name이 JSVReqSeq이면 Request Sequence이다.
%m	HTTP Request 메소드를 표시한다.
%p	Request가 도착한 서버의 포트 번호를 표시한다.
%q	HTTP Request의 query 값을 표시한다.
%r	HTTP Request의 Request line 전체를 표시한다.
%R	HTTP Request의 Request line 전체를 표시한다. CheckURL이나 URLRewrite 기능에 의해 변경된 Request line을 표시한다.
%s	응답에 사용된 HTTP Status Code를 표시한다.
%t	요청처리를 마친 시간을 표시한다.

포맷	설명
%T	Request를 처리하는 데 소요된 시간을 표시한다. (단위: 초)
%u	HTTP 인증에 사용된 user 이름을 표시한다.
%U	HTTP Request URI를 표시한다.
%v	Host Header 필드 값을 표시한다.
%z	응답이 압축된 경우 압축 전/후의 응답 크기와 압축률을 표시한다.

CLF에 해당되는 "%h %l %u %t \"%r\" %s %b"는 "default"라는 이름으로 제공한다. 위의 예에서 보면 액세스 로그는 "default"로 설정되어 있으므로 실제로 적용되는 Log Format은 CLF를 준수하는 형태로 로그에 기록된다.

또한, %d 옵션과 함께 { }안에 strftime 함수에서 제공하는 날짜 및 시간의 여러 가지 형식을 지정할 수 있다.
기본값은 %t와 동일하게 "[13/Feb/2001:16:47:24 +0900]"와 같은 형태로 기록된다.

지시자를 사용하여 다음과 같은 형태의 Format을 만들어 낼 수 있다.

```
custom_log1
    Format = "%h %l %u %t \"%r\" %s %b",
    Filename = "log/custom1.log"
custom_log2
    Format = "[%{%Y.%m.%d %H:%M:%S}d] %a \"%r\" %s %T",
    Filename = "log/custom2.log"
```

4.4. Tmax 연동

WebtoB는 aps 라이브러리를 이용하여 TP-Motinor 제품 Tmax와 연동이 가능하다. Tmax와 연동을 위해서 Tmax를 별도로 설치해야 한다.

WebtoB에서는 WBAPI를 이용하여 클라이언트 프로그램만 작성하며 서버 프로그램은 Tmax에 있는 것을 사용한다. 즉, WebtoB의 클라이언트 프로그램에서 Tmax의 서버 프로그램으로 요청을 보내고 그 응답을 받아와서 WebtoB의 클라이언트(웹 브라우저)로 결과에 대한 응답을 줄 수 있다.

4.4.1. Tmax 연동을 위한 환경설정

Tmax와 연동은 WBAPI를 이용하는 것이므로 환경 파일 작성 방법 역시 그와 동일하다.

다음은 환경 파일의 예이다.

<apsl.m>

```
*DOMAIN
webtob_apsl

*NODE
tmaxi2
    WebtoBDir="$WEBTOBDIR",
    SHMKEY = 74125,
    Docroot = "docs/",
```

```
    Port = "8797",
    HTH = 1,
    Logging = "log1",
    AppDir = "ap/",
    ErrorLog = "log2"

*SVRGROUP
htmlg    NodeName = "tmaxi2", SvrType = HTML
webapg  NodeName = "tmaxi2", SvrType = WEBSTD

*SERVER
wbsvrinit  SvgName = webapg, MinProc = 10, MaxProc = 10
wbquery    SvgName = webapg, MinProc = 10, MaxProc = 10
wbsession  SvgName = webapg, MinProc = 10, MaxProc = 10

*SERVICE
test       SvrName = wbsvrinit
query      SvrName = wbquery
wbsession  SvrName = wbsession

*URI
uri1      Uri = "/svct/", SvrType = WEBSTD

*LOGGING
log1
    Format = "default",
    Filename = "log/access.log",
    Option="Sync"
log2
    Format = "",
    Filename = "log/error.log",
    Option="Sync"
```

4.4.2. 사용예

위에 제시한 apsl.m에 해당하는 클라이언트 프로그램과 사용에 대한 내용은 Tmax 연동을 참고한다.

4.5. JEUS 연동

WebtoB는 차세대 웹 애플리케이션 서버(Web Application Server : WAS)인 JEUS와 연동되어 일반 웹 서버에서 제공하지 못하는 기능들을 구현할 수 있다. 일반적인 웹 서비스를 포함한 전자상거래에서는 보안과 트랜잭션 등이 반드시 보장되어야 한다. 또한 대용량의 데이터를 다루는 일이나 사용자의 관리 등의 측면에서 웹 애플리케이션 서버인 JEUS를 이용하는 것이 바람직하다.

설정 방법에 있어서 WebtoB Base 이상에서 제공되는 JEUS와의 연동 방법과 WebtoB Standard에서 제공되는 내장 Servlet Engine과의 연동 방법은 JEUS 버전별 항목 이름에만 다소 차이가 있으며 기본 원리는 동일하다.

WebtoB와 JEUS 연동을 위해 설정해야 할 파일은 다음과 같다.

- WebtoB 환경 파일 (예: http.m)
- JEUS 환경 파일 : WEBMain.xml

WebtoB와 JEUS 연동을 위해 서로의 값을 똑같이 설정해야 하는 항목은 다음과 같다.

- WebtoB-JEUS 간 연결 포트 번호
- HTH 수
- 서버 이름(WebtoB의 JSV 설정 서버이름과 JEUS의 Registration ID)
- 프로세스 수(WebtoB의 MinProc/MaxProc와 JEUS의 Thread-pool 수)

4.5.1. WebtoB와 JEUS 연결의 특징

WebtoB와 JEUS의 연결은 JEUS가 WebtoB에게 연결하는 구조로 되어 있다. WebtoB는 외부 네트워크에서 서비스하고, JEUS는 내부 네트워크에서 운영할 경우 유용하다. 그 이유는 내부 네트워크의 방화벽에서 in-bound로 연결되는 것을 엄격히 관리하지만, out-bound 연결은 관대하게 관리하기 때문이다.

JEUS가 WebtoB에 연결하는 경우 WebtoB가 먼저 기동하고 WebtoB가 JEUS의 연결을 기다리고 있는 상태에서 JEUS를 기동할 것을 권장한다. JEUS는 기동하면서 WebtoB와 연결하게 되는데, 만약 WebtoB가 기동되지 않은 경우 주기적으로 연결을 재시도된다. JEUS가 연결을 재시도할 때 마다 JEUS의 로그에 에러 메시지가 기록된다. 연결에 성공하면 JEUS의 로그에 연결이 성공되었다는 메시지가 저장되고, 이후 Servlet이나 JSP 등 JEUS의 Servlet Engine을 이용해야 하는 사용자의 요구는 이 연결을 통해서 JEUS에 전달되어 처리된다. 그리고 WebtoB는 그 결과를 받아 클라이언트에게 전송해 준다.

WebtoB와 JEUS와의 연결은 persistent connection이므로 중간에 네트워크 오류나 서버의 문제가 생기지 않는 이상은 연결을 끊지 않는다.

4.5.2. JEUS 6 연동 설정(Base+)

비교적 간단한 비즈니스 로직을 수행하는 중소규모의 온라인 서비스를 제공하고자 할 경우에는 WebtoB Servlet Engine과의 연동만으로도 서비스가 가능하지만, 대규모의 전자상거래 시스템을 구축하려 한다면 WebtoB와 JEUS을 연동하여 이용하는 것을 권장한다.

본 절에서는 JEUS 6 이상과의 연동을 기준으로 설명한다. JEUS 7 연동은 "4.5.3. 내장 Servlet Engine 연동 설정(Enterprise)" 내용을 참고한다.

WebtoB 환경 파일 설정

- **NODE 절**

  다음과 같이 HTH 개수와 JEUS와의 연결을 맺을 JsvPort를 설정한다.

```
*NODE
          ...
        HTH = 1,
        JsvPort = 9999,
          ...
```

  HTH에 설정된 값은 WebtoB에서 HTH 프로세스의 개수로서 이 값과 JEUS 환경 파일 중 WEBMain.xml의 <webtob-listener> 하위에 있는 <hth-count> 항목 값과 일치해야 한다. 또한 JsvPort는 웹 컨테이너와 연결을 맺을 포트 번호로서 실제 웹 브라우저로 요청받는 포트 번호와는 무관하다. JEUS 환경 파일 WEBMain.xml의

<webtob-listener>하위에 있는 <port> 항목 값과 일치해야 한다.

- **SVRGROUP 절**

  SVRTYPE이 JSV인 서버 그룹을 설정한다.

```
*SVRGROUP
jsvg       SvrType = JSV
```

- **SERVER 절**

  서비스와 실제로 웹 컨테이너와 연결하여 작업을 할 서버를 설정한다.

```
*SERVER
MyGroup    SvgName = jsvg, MinProc = 4, MaxProc = 10
```

  서버의 이름으로 MyGroup이고, 서버 그룹의 이름은 SVRGROUP 부분에서 설정한 jsvg이다.

  MinProc은 웹 컨테이너와의 최소 연결 개수이고, MaxProc은 웹 컨테이너와의 최대 연결 개수이다. 이 개수들은 JEUS 환경 파일 중 WEBMain.xml의 <webtob-listener> 하위에 있는 <thread-pool> 설정값과 각각 일치하거나 커야 한다.

- **URI 절**

  URI 부분은 어떤 URI가 요청이 되었을 때 어떤 서버를 수행시킬지를 지정하는 것으로 서버(MyGroup)의 서비스에 대한 URI를 설정한다.

  다음은 "/examples/" URI에 대해 JSV 서버(MyGroup)의 서비스를 수행하도록 설정하는 예이다.

```
*URI
uri1     Uri = "/examples/",
         SvrType = JSV,
         SvrName = MyGroup
```

JEUS 6 환경 파일 설정

WebtoB 연결 설정을 위해 아래 경로에 있는 **WEBMain.xml**을 수정해야 한다.

```
$JEUS_HOME/config/<node_name>/<node_name>_servlet_<engine_name>/
```

<WEBMain.xml>

```
<webtob-listener>
  <listener-id>webtob1</listener-id>
  <port>9999</port>
  <webtob-address>localhost</webtob-address>
  <registration-id>MyGroup</registration-id>
```

```
  <hth-count>1</hth-count>
  <thread-pool>
    <min>10</min>
    <max>10</max>
    <step>0</step>
    <max-idle-time>30000</max-idle-time>
  </thread-pool>
</webtob-listener>
```

WEBMain.xml 파일에서 확인하고 수정해야 할 항목은 <webserver-connection> 하위에 있는 <webtob-listener> 하위 항목들이며 각 항목에 대한 설명은 다음과 같다.

태그	설명
<port>	WebtoB Servlet Engine 웹 컨테이너와 WebtoB와의 연결을 맺을 포트 번호를 설정하는 것으로 WebtoB 설정의 NODE 절의 JsvPort 값과 일치해야 한다.
<webtob-address>	연결을 맺을 WebtoB의 IP 주소를 설정한다.
<registration-id>	WebtoB와 처음 연결을 맺을 때 등록과정 중 사용할 등록 ID로 WebtoB SERVER 설정 중 연결하고자 하는 서버 이름(예: MyGroup)과 같아야 한다.
<hth-count>	WebtoB의 NODE 절의 HTH와 같은 값을 설정한다. 실제 연결은 WebtoB의 각 HTH에게 연결하기 때문에 이 값이 일치해야 한다.
<thread-pool>, <min>, <max>	Thread Pool에 최소한 유지되어야 하는 Thread의 개수와 최대 개수를 정의하는 것으로 WebtoB 환경 파일에 정의한 MyGroup 서버에 대한 MinProc/MaxProc 값과 일치하거나 적어야 한다.

하나의 컨텍스트 그룹은 여러 개의 웹 서버 연결을 가질 수 있는데 각각의 연결마다 포트 번호가 달라야 한다. 이렇게 여러 개의 웹 서버 연결을 가지는 것은 웹 서버가 여러 노드에 존재할 때나 다른 종류의 웹 서버로부터 요청을 받을 때에 사용한다.

Servlet Engine 기동 및 종료

명령어 모드에서 다음의 단계별로 WebtoB와 WebtoB Servlet Engine을 기동시키도록 한다.

1. WebtoB 환경 파일을 컴파일한다.

   ```
   C:\> wscfl –i http.m
   ```

2. WebtoB를 기동시킨다.

   ```
   C:\>wsboot
   ```

3. 아래 명령어를 사용하여 JEUS Manager를 시작한다.

   ```
   C:\>jeus
   ```

4. JEUS를 부팅하기 위해서 인증 과정을 거쳐 jeusadmin 콘솔 툴을 이용한다.

```
C:\>jeusadmin webmain
Login name>administrator
Password>
JEUS 6.0 Jeus Manager Controller
webmain>
```

webmain은 JEUS를 시작한 노드로써 해당 노드에 대해 권한이 있는 UserName과 그에 따른 Password 인증 과정을 거쳐야 jeusadmin 명령어를 이용할 수 있다.

ID는 기본으로 adminstrator이고, Password 값은 설치 과정에서 입력한 값을 기억하여 입력한다.

5. 다음과 같이 JEUS를 부팅하여 WebtoB와 연동시키므로써 원하는 서비스를 이용하게 된다.

```
webmain> boot
```

부팅 과정은 2단계로 이루어진다.

a. JEUS 서비스를 시작한 후 각각의 엔진을 시작한다.

b. JEUS 서비스 시작 과정을 살펴보면 위 명령은 JEUS Manager로 전달된다.

6. JEUS 서비스와 엔진을 종료하고 싶으면 다음 명령을 수행한다.

```
webmain> down
```

7. 다음과 같은 명령어를 이용해 JEUS 서버를 종료한다.

```
webmain>jeusexit
```

예제

WebtoB와 JEUS 연동 환경설정 파일을 참고한다.

4.5.3. 내장 Servlet Engine(JEUS 7) 연동 설정(Enterprise)

WebtoB Enterprise 이상부터 제공되는 Servlet Engine 연동을 위한 설정은 앞서 설명한 JEUS와의 연동 설정 방법과 동일하다. 단, 현재 Servlet Engine으로 제공되는 버전은 JEUS 7에 해당하므로 앞서 설명한 JEUS 6 이상에서의 설정과는 항목 이름에서 약간의 차이가 있다.

WebtoB 환경 파일 설정

JEUS 6 연동 설정(Base+)의 “WebtoB 환경 파일 설정”의 방법과 동일하다.

내장 Servlet Engine 환경 파일 설정

WebtoB와 연결 설정을 위해서는 아래 경로에 있는 **domain.xml**을 수정해야 한다.

```
${WEBTOBDIR}/jeus/domains/domain1/config/domain.xml
```

<domain.xml>

```
<webtob-connector>
  <name>webtob</name>
  <network-address>
     <port>9999</port>
     <ip-address>localhost</ip-address>
  </network-address>
  <hth-count>1</hth-count>
  <thread-pool>
     <number>10</number>
  </thread-pool>
  <registration-id>MyGroup</registration-id>
  <output-buffer-size>8192</output-buffer-size>
  <reconnect-interval>5000</reconnect-interval>
</webtob-connector>
```

domain.xml 파일에서 확인하고 수정해야 할 항목은 <webtob-connector> 하위에 있는 항목들이며 각 항목에 대한 설명은 다음과 같다.

태그	설명
<network-address>, <port>	WebtoB Servlet Engine이 WebtoB와의 연결을 맺을 포트 번호를 설정하는 것으로 WebtoB 설정 중 NODE 절의 JsvPort 값과 일치해야 한다.
<network-address>, <ip-address>	연결을 맺을 WebtoB의 IP 주소를 지정한다.
<hth-count>	WebtoB의 NODE 절의 HTH와 같은 값을 설정한다. 실제 연결은 WebtoB의 각 HTH에게 연결하기 때문에 이 값이 일치해야 한다.
<thread-pool>, <number>	Thread Pool에 유지되어야 하는 Thread의 개수를 정의하는 것으로 WebtoB 환경 파일에 정의한 MyGroup 서버에 대한 MaxProc 값과 일치하거나 적어야 한다.
<registration-id>	WebtoB와 처음 연결을 맺을 때 등록 과정 중 사용할 등록 ID로 WebtoB 환경 파일에서 SERVER 부분의 서버 이름(MyGroup)과 같아야 한다.

내장 Servlet Engine 기동 및 종료

명령어 모드에서 다음의 단계별로 WebtoB와 WebtoB 내장 Servlet Engine을 기동시키도록 한다. 자세한 설정 방법은 JEUS 7 Fix#4의 매뉴얼을 참고하도록 한다.

1. WebtoB 환경 파일을 컴파일한다.

```
C:\> wscfl –i http.m
```

2. WebtoB를 기동시킨다.

```
C:\>wsboot
```

3. 아래 명령어를 사용하여 JEUS의 DAS를 기동한다.

```
C:\>startDomainAdminServer -u administrator -p <password>
```

4. 아래 명령어를 사용하여 JEUS의 MS를 기동한다.

```
C:\>startManagedServer -domain jeus_domain -server server1 -u administrator -p <password>
```

   JEUS MS가 정상적으로 부트되면 examples가 deploy되는데 이를 확인하기 위해서는 웹 브라우저에서 "http://localhost:8088/examples/"을 호출한다.

5. 다음은 jeusadmin에 접속하는 방법이다.

```
C:\>jeusadmin -u administrator -p <password>
Attempting to connect to 127.0.0.1:9736.
The connection has been established to Domain Administration Server adminServer in the domain
jeus_domain.
JEUS7 Administration Tool
To view help, use the 'help' command.
[DAS]jeus_domain.adminServer>
```

   JEUS 관리자의 사용자 이름과 패스워드를 입력한다. 일반적으로 관리자의 계정은 'administrator'이고 패스워드는 Servlet Engine을 설치할 때 입력한 값이다. 잠시 후 프롬프트가 다시 뜨면 JEUS가 제대로 기동되었고, 다시 명령어를 받을 수 있는 상태가 되었다는 것을 나타낸다.

6. WebAdmin을 통해서 Servlet Engine을 관리할 수 있다.

```
http://localhost:9736/webadmin
```

   웹 브라우저를 열어서 주소 창에 위와 같이 주소를 입력한다. JEUS 관리자의 계정과 패스워드를 입력하고 **[Login]** 버튼을 클릭한다. 일반적으로 관리자의 계정은 'administrator'이고 패스워드는 JEUS를 설치할 때 입력한 값이다.

7. JEUS 서비스와 엔진을 종료하고 싶으면 다음 명령을 수행한다.

```
[DAS]jeus_domain.adminServer>local-shutdown
```

8. 다음과 같은 명령어를 이용해 jeusadmin을 종료한다.

```
exit
```

예제

WebtoB와 JEUS 연동 환경설정 파일을 참조한다.

4.6. 다른 WAS 연동

WebtoB는 Reverse Proxy를 이용하여 JEUS뿐만 아니라 타 WAS와 연동이 가능하다. WAS는 일반적으로 HTTP Listener를 가지고 있는데, WebtoB는 reverse proxy 설정을 통해 WAS의 HTTP Listener와 연결을 맺어 연동하게 된다.

WebtoB Enterprise 이상에서 제공되는 Reverse Proxy Group 설정을 통해 다중 WAS 구성이 가능하다.

WebtoB와 WAS 연동을 위해 설정해야 할 파일은 다음과 같다.

- WebtoB 환경 파일 (예: http.m)
- WAS 환경 파일 (예: WEBMain.xml)

WebtoB와 WAS 연동을 위해 WAS에서 확인해야할 내용은 다음과 같다. (여기에서는 타 WAS의 설정을 다루지 않으며, 필요한 경우 타 WAS의 매뉴얼을 참고하도록 한다.)

- WebtoB가 WAS로 연결할 HTTP Listener 설정(Listen IP, Port, thread 수)
- WebtoB와 WAS의 연결을 persistent connection으로 유지하고자 하는 경우 PING 체크를 위한 application 설정
- Sticky Session routing을 사용하고자 하는 경우 routing id
- WebtoB와 WAS간 SSL 암호화 통신을 하고자 하는 경우 HTTP Listener에 대한 SSL 설정

4.6.1. WebtoB의 Reverse Proxy를 통한 WAS 연결의 특징

WebtoB와 WAS의 연결은 WebtoB가 Reverse Proxy 설정을 통해 WAS에게 연결하는 구조이다. WebtoB는 외부 네트워크에서 서비스하고, WAS는 내부 네트워크에서 운영할 경우에도 유용하다. 외부 네트워크에서 서비스 하는 WebtoB가 내부 서버(WAS)로부터 받은 응답을 rewriting할 수 있기 때문이다.

WebtoB는 클라이언트로부터 HTTP 요청을 받은 시점에 WAS로 연결을 시도한다. 만약 WAS가 기동되지 않았다면 클라이언트에게 503 에러를 응답하게 된다. 연결에 성공하면 WebtoB는 이 연결을 통해서 사용자의 요청을 WAS에게 전달하고그 결과를 받아 클라이언트에게 전송해 준다.

WebtoB는 WAS와의 연결을 persistent connection으로 설정하여 재사용할 수 있는데 이를 위해서는 WebtoB와 WAS의 PING 체크를 위한 설정이 필요하다. Min, Max를 설정한 경우 클라이언트의 동시 요청량에 따라 동적으로 connection을 관리한다. 또한, WebtoB와 WAS는 SSL 통신이 가능하다.

4.6.2. 다중 WAS 연동 설정(Enterprise+)

다중 WAS를 구성하고자 하는 경우 WebtoB Enterprise에서 제공되는 Reverse Proxy Group 설정을 사용하면 되고, 하나의 WAS만 연동하고자 한다면 Reverse Proxy 설정만 사용하면 된다.

WebtoB 환경 파일 설정

- **REVERSE_PROXY_GROUP 절**

  다음과 같이 Reverse Proxy를 그룹으로 묶어서 하나로 관리할 항목들을 설정한다.

```
*REVERSE_PROXY_GROUP
rproxyG1
         VhostName = "vhost1",
         PathPrefix = "/before/",
         RegExp = "\.(do|jsp)$",
         ServerPathPrefix = "/after/",
         RewriteHtmlUrl = "/after/ /before/",
         HtmlUrl = "a href",
         HtmlUrl = "img src",
         HtmlUrl = "link href",
         HtmlUrl = "script src",
         HtmlUrl = "frame src",
         HtmlUrl = "td background",
         HtmlUrl = "table background",
         HtmlUrl = "iframe src",
         HtmlUrl = "OBJECT CODEBASE",
         RewriteHtmlMaxSize = 4194304,
         WBRoutingCookieKey = "W2BRID"
        #SessionIDCookieKey = "JSESSIONID"
```

  위 설정은 "/before/"로 시작하는 요청을 "/after/"로 변경하여 서비스하기 위한 설정이 된다.

  다중 WAS를 구성하는 것은 각각의 WAS에서 동일한 서비스를 하기 위한 목적이기 때문에 VhostName, PathPrefix, RegExp, ServerPathPrefix 항목은 Reverse Proxy Group과 Reverse Proxy가 동일하게 설정하거나 Reverse Proxy Group에만 설정해야 한다.

  WAS의 Sticky Session routingid와 상관없이 자체 routingid를 사용하고자 하는 경우 WBRoutingCookieKey 설정을 사용하면 된다.

- **REVERSE_PROXY 절**

  Reverse Proxy Group명과 WAS의 HTTP Listener 정보(Ip:Port) 및 각 WAS별 연동정보를 설정한다.

```
*REVERSE_PROXY
rproxy1
         ServerAddress = "127.0.0.1:8088",
         ReverseProxyGroupName = "rproxyG1",
         MinPersistentServerConnections = 1,
         MaxPersistentServerConnections = 20,
         PersistentServerCheckTime = 50,
         PersistentServerTimeout = 300,
```

```
         PersistentServerCheckUrl = "/after/ping.html",
         MaxWebSocketConnections = 20
        #StickySessionRoutingID = "was1_servlet_engine",
        #ProxySSLFlag = Y,
        #ProxySSLName = pssl1

rproxy2
         ServerAddress = "127.0.0.1:8089",
         ReverseProxyGroupName = "rproxyG1",
         MinPersistentServerConnections = 1,
         MaxPersistentServerConnections = 20,
         PersistentServerCheckTime = 50,
         PersistentServerTimeout = 300,
         PersistentServerCheckUrl = "/after/ping.html",
         MaxWebSocketConnections = 20
        #StickySessionRoutingID = "was2_servlet_engine",
        #ProxySSLFlag = Y,
        #ProxySSLName = pssl1
```

위 설정은 rproxyG1 그룹에 속하는 rproxy1(127.0.0.1:8088), rproxy2(127.0.0.1:8089) 설정을 한 경우이며, 요청은 RR(Round Robin) 방식으로 분배된다. 하나의 Reverse Proxy 내에서 연결이 여러개 존재할 때 요청은 각 연결에 대하여 FA(First Assign) 방식으로 분배된다.

MaxPersistentServerConnections은 동시 요청을 최대로 처리할 수 있는 연결 수를 설정한다. MaxPersistentServerConnections보다 많은 요청이 동시에 들어온다면 WebtoB가 큐잉하게 된다.

MinPersistentServerConnections은 요청이 없더라도 지속적인 연결을 유지하고자 하는 연결 수를 설정한다. MinPersistentServerConnections을 넘어선 연결들은 PersistentServerTimeout 시간 동안 어떠한 요청도 들어오지 않으면 끊어지게 된다.

PersistentServerCheckTime은 WAS와의 연결을 지속적으로 유지하기 위해 PING 메시지를 보내는 주기이다. 이 값은 WAS의 HTTP Listener에 대한 KeepAliveTimeout보다 작게 설정해야 한다. 그렇지 않으면 WAS에서 먼저 연결을 끊어버릴 것이기 때문이다. 해당 주기마다 WebtoB는 WAS에 주기적으로 PING 메시지를 보내고 이에 대한 응답을 받음으로써 각 연결들을 지속적으로 유지하게 된다. 이를 바탕으로 WebtoB는 WAS와의 연결에 대하여 connection pool을 구성하게 되고 각 connection을 동적으로 관리할 수 있게 된다.

PING 메시지는 PersistentServerCheckUrl 설정을 가지고 만들어진 HTTP HEAD Request이다. 위와 같이 설정하게 되면 PING 메시지는 "HEAD /after/ping.html HTTP/1.1"가 된다. HEAD 메소드는 HTTP Response Header만 받고자 하는 메소드이기 때문에 PING 메시지에 대한 응답(PONG 메시지)을 최소화 하는데 유용하다. 이를 위해 WAS에서는 반드시 "/after/ping.html" application을 준비해야 한다.

WAS와의 connection을 WebSocket으로 사용하는 경우 MaxWebSocketConnections 설정을 통해 WebSocket으로 사용하고자 하는 connection 수를 제한할 수 있다.

Reverse Proxy Group에 WBRoutingCookieKey를 설정한 경우 rproxy1, rproxy2에 대해 자체 routing하게 된다. rproxy1에서 처리된 응답에 대해 WBRoutingCookieKey와 rproxy1을 Base64로 인코딩된 값을 Set-Cookie 헤더 추가해준다. 이후 동일 클라이언트로부터의 요청을 받은 경우 Cookie의 해당 값을 이용하여 rproxy1로 요청을 routing하게 된다. WBRoutingCookieKey를 사용하지 않는 경우에 StickySessionRoutingID는 Reverse Proxy Group의 SessionIDCookieKey설정과 함께 Sticky Session routing을 사용하는 경우 설정한다. WAS에서 "Set-Cookie" Responose Header에 JSESSIONID로 만들어 주는 값의 dot(.)이후의 엔진명을 넣어주면 된다.

ProxySSLFlag, ProxySSLName은 WebtoB와 WAS의 연결에 대해 SSL 암호화 통신을 할 경우 설정한다.

- **PROXY_SSL 절**

  WebtoB와 WAS의 SSL 암호화 통신을 할 경우 설정한다.

```
*PROXY_SSL
pssl1
         Verify = 2,
         VerifyDepth = 1,
         RequiredCiphers = "ALL:!ADH:!EXPORT56:RC4+RSA:!SHA:+HIGH:
+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL",
         CACertificateFile="$(WEBTOBDIR)/ssl/server.crt",
         CheckPeerValidPeriod = Y
```

  위 설정은 WebtoB가 WAS로 연결을 맺을 때 SSL 연결을 위한 설정이다. WebtoB가 클라이언트가 되고 WAS가 서버가 되기 때문에 WAS는 서버를 위한 SSL 설정을 해야 한다. 이렇게 설정된 경우 WebtoB는 WAS로 연결을 맺을 때 TCP 연결 이후 SSL Handshake 과정을 요청한다.

  SSL Handshake 과정에서 서버 인증서를 검증하고자 하는 경우 Verify를 2로 설정하고 그렇지 않다면 0으로 설정하면 된다. 서버 인증서를 검증하는 경우 반드시 CACertificateFile을 설정해야 한다.

설정, 통계 및 connection 상태 정보 확인

wsadmin 모드에서 Reverse Proxy Group, Reverse Proxy에 대한 설정, 통계 및 각 연결에 대한 상태 정보를 확인한다.

1. Reverse Proxy Group 설정 확인

```
wsadmin> cfg -rpg
    REVERSE_PROXY_GROPUP(0): Name = rproxyG1,
         VhostName = "vhost1",
         PathPrefix = "/before/",
         ServerPathPrefix = "/after/",
         RewriteHtmlMaxSize = 4194304,
         RewriteHtmlUrl = "/after/ /before/",
         HtmlUrl = "a href",
         HtmlUrl = "img src",
         HtmlUrl = "link href",
         HtmlUrl = "script src",
         HtmlUrl = "frame src",
         HtmlUrl = "td background",
         HtmlUrl = "table background",
         HtmlUrl = "iframe src",
         HtmlUrl = "OBJECT CODEBASE",
         SessionIdCookieKey = "JSESSIONID",
         FlexibleStickySessionRouting = N       # (N:0, Y:1),
         ReverseProxyEntries = 2
```

2. Reverse Proxy 설정 확인

```
wsadmin> cfg -rp
    REVERSE_PROXY(0): Name = rproxy1,
         ReverseProxyGroupName = "rproxyG1",
         VhostName = "vhost1",
         PathPrefix = "/before/",
         ServerPathPrefix = "/after/",
         ServerAddress = "127.0.0.1:8088",
         HttpInBufSize = 16384,
         RewriteHtmlMaxSize = 4194304,
         RewriteHtmlUrl = "/after/ /before/",
         HtmlUrl = "a href",
         HtmlUrl = "img src",
         HtmlUrl = "link href",
         HtmlUrl = "script src",
         HtmlUrl = "frame src",
         HtmlUrl = "td background",
         HtmlUrl = "table background",
         HtmlUrl = "iframe src",
         HtmlUrl = "OBJECT CODEBASE",
         MinPersistentServerConnections = 1,
         MaxPersistentServerConnections = 20,
         PersistentServerCheckTime = 50,
         PersistentServerTimeout = 300,
         PersistentServerCheckUrl = "/after/ping.html",
         ProxySslFlag = N
    REVERSE_PROXY(1): Name = rproxy2,
         ReverseProxyGroupName = "rproxyG1",
         VhostName = "vhost1",
         PathPrefix = "/before/",
         ServerPathPrefix = "/after/",
         ServerAddress = "127.0.0.1:8089",
         HttpInBufSize = 16384,
         RewriteHtmlMaxSize = 4194304,
         RewriteHtmlUrl = "/after/ /before/",
         HtmlUrl = "a href",
         HtmlUrl = "img src",
         HtmlUrl = "link href",
         HtmlUrl = "script src",
         HtmlUrl = "frame src",
         HtmlUrl = "td background",
         HtmlUrl = "table background",
         HtmlUrl = "iframe src",
         HtmlUrl = "OBJECT CODEBASE",
         MinPersistentServerConnections = 1,
         MaxPersistentServerConnections = 20,
         PersistentServerCheckTime = 50,
         PersistentServerTimeout = 300,
         PersistentServerCheckUrl = "/after/ping.html",
         ProxySslFlag = N
```

3. Proxy SSL 설정 확인

```
wsadmin> cfg -pssl
    PROXY_SSL(0): Name = pssl1,
         Verify= 2,
         VerifyDepth = 1,
         CheckPeerValidPeriod = Y       # (N:0, Y:1),
```

```
        CACertificatePath = "/home/tmax/server/webtob/ssl/",
        CACertificateFile = "/home/tmax/server/webtob/ssl/server.crt",
        Protocols = "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3",
        RequiredCiphers = "ALL:!ADH:!EXPORT56:RC4+RSA:!SHA:+HIGH:
+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL"
```

4. Reverse Proxy Group 단위의 통계 및 연결 정보 확인

```
wsadmin> st -rpg

-------------------------------------------------------------------------------
 hth  (rpgi)rpgname  rproxyname     count(qcnt)    avg cons   remote_ipaddr:port
-------------------------------------------------------------------------------
   0  ( 0)rproxyG1   rproxy1            1(   0) 0.0025    1        127.0.0.1:8088
   0  ( 0)rproxyG1   rproxy2            1(   0) 0.0021    1        127.0.0.1:8089
```

다음은 출력 항목에 대한 설명이다.

항목	설명
count	요청 처리 수
qcnt	Reverse Proxy Group 단위의 큐잉된 요청 수(위와 같은 경우 rproxy1, rproxy2의 qcnt는 동일)
avg	평균 처리시간(초)
cons	WAS와 연결된 connection 수

5. Reverse Proxy별 connection 단위의 통계 및 상태 정보 확인

```
wsadmin> st -rp

HTH 0: RDY
 -------------------------------------------------------------------------------
 (rpi)rproxy_name  rpgname     status     count  idle spri  clid  ssl websocket
 -------------------------------------------------------------------------------
 (  0)rproxy1      rproxyG1       RDY         1    14    1    -1   N      N
 (  1)rproxy2      rproxyG1       RDY         1    10    2    -1   N      N
```

다음은 출력 항목에 대한 설명이다.

항목	설명
status	해당 connection의 상태 정보
count	요청 처리 수
idle	idle 시간(초)
spri	connection에 대한 내부 관리 index
clid	요청 처리 중인 클라이언트 index
ssl	WAS와의 SSL 사용 여부
websocket	connection이 websocket으로 사용되고 있는지 여부

4.7. URLRewrite

Apache의 mod_rewrite를 WebtoB에 포팅한 기능으로 rule-based URL rewriting 기능을 제공한다.

4.7.1. WebtoB 설정

URLRewrite 기능을 사용하기 위해서는 NODE 절의 'URLRewrite' 항목은 'Y'로 설정되어야 하고 URLRewriteConfig 항목에 Condition과 Rule에 관련된 설정을 해야 한다.

다음은 URLRewrite 기능을 사용하기 위한 NODE 절 설정에 대한 예제이다.

```
*NODE
mynode
    URLRewrite = Y,
    URLRewriteConfig = "${WEBTOBDIR}/config/rewrite.conf"
```

4.7.2. URLRewriteConfig 파일 설정

URLRewriteConfig에는 Apache의 mod_rewrite의 기능 중 RewriteCond와 RewriteRule 설정을 사용할 수 있다. 모든 설정을 사용할 수 있는 것은 아니며, 뒤에서 설명할 몇 가지 기능은 사용할 수 없다.

다음에 설명하는 URLRewriteConfig 설정 내용은 "Apache 2.2 mod_rewrite"를 참고하여 작성한 내용이다. 설명된 내용 중 WebtoB에서 동작하지 않는 기능이 있을 수 있으니 유의해야 한다.

4.7.3. RewriteCond

RewriteCond는 rewriting 조건을 명시하며, 다음과 같은 형식으로 설정한다. TestString과 CondPattern을 매칭하여 조건이 맞을 경우 RewriteRule 설정대로 해당되는 패턴을 교체한다.

```
RewriteCond <TestString> <CondPattern> [flags]
```

- <TestString>

  TestString에는 다음과 같은 예약어와 일반 문자열을 사용할 수 있다.

  - $N (0 <= N <= 9)
    - RewriteRule의 Pattern 중 괄호로 묶인 패턴을 참조한다.
  - %N (1 <= N <= 9)
    - RewriteCond의 CondPattern 중 괄호로 묶인 패턴을 참조한다.
    - $N과 %N은 다음과 같은 Regex Back-Reference 구조를 가진다.

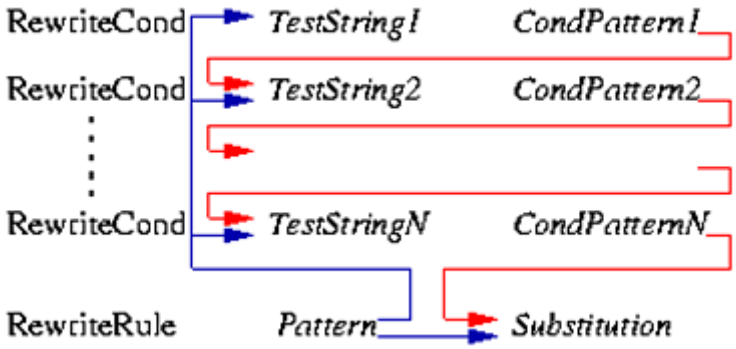


그림 4. Regex Back-Reference

◦ %{SERVER_VARIABLE_NAME}

CGI에서 사용하는 서버 환경변수 중 일부와 몇 가지 추가적인 변수를 참조한다.

- %{ENV:variable}은 환경변수를 참조한다.
- %{HTTP:header}는 HTTP Request Header를 참조한다.
- 이외에도 다음과 같은 변수를 사용할 수 있다.

변수	설명
HTTP headers	HTTP_USER_AGENT, HTTP_REFERER, HTTP_COOKIE, HTTP_FORWARDED, HTTP_HOST, HTTP_PROXY_CONNECTION, HTTP_ACCEPT
connection & request	REMOTE_ADDR, REMOTE_PORT, REMOTE_USER, REMOTE_METHOD, QUERY_STRING, AUTH_TYPE
server internals	DOCUMENT_ROOT, SERVER_NAME, SERVER_ADDR, SERVER_PORT, SERVER_PROTOCOL, SERVER_SOFTWARE
date and time	TIME_YEAR, TIME_MON, TIME_DAY, TIME_HOUR, TIME_SEC, TIME_WDAY, TIME
specials	THE_REQUEST, REQUEST_URI, HTTPS

Apache의 mod_rewrite에서 제공하는 %{SSL:variable}, %{LA-U:variable}, %{LA-F:variable},

REMOTE_HOST, REMOTE_IDENT, SCRIPT_FILENAME, SCRIPT_USER, SCRIPT_GROUP,

SERVER_ADMIN, API_VERSION, REQUEST_FILENAME, IS_SUBREQ는 사용할 수 없다.

- <CondPattern>

CondPattern은 Perl compatible regular expression을 사용할 수 있으며, 다음과 같은 추가적인 패턴을 사용할 수 있다.

지시자	설명
!Pattern	Pattern에 매칭되지 않는 경우를 지정한다.
<CondPattern	TestString이 CondPattern보다 사전순서상 앞에 있을 경우를 매칭한다.
>CondPattern	TestString이 CondPattern보다 사전순서상 뒤에 있을 경우를 매칭한다.
=CondPattern	TestString이 CondPattern과 일치하는 경우를 매칭한다.
-d	TestString을 path로 간주하여 디렉터리일 경우를 매칭한다.
-f	TestString을 path로 간주하여 파일일 경우를 매칭한다.
-s	TestString을 path로 간주하여 파일 사이즈가 0보다 큰 경우를 매칭한다.
-l	TestString을 path로 간주하여 symbolic link일 경우를 매칭한다.
-x	TestString을 path로 간주하여 실행 파일일 경우를 매칭한다.
-F	TestString을 path로 간주하여 서버 설정상 접근이 가능한 경우를 매칭한다.
-U	TestString을 URL로 간주하여 서버 설정상 접근이 가능한 경우를 매칭한다.

모든 패턴에는 '!'을 붙여서 반대 의미로 매칭할 수 있다.

참고로 다음은 Regex vocabulary에 대한 설명이다.

지시자	설명
.	matches any single character
+	repeats the previous match one or more times
*	repeats the previous match zero or more times
?	makes the match optional
^	matches the beginning of the string
$	matches the end of the string
( )	groups several characters into a single unit, and captures a match for use in backreference
[ ]	a character class - matches one of the characters
[^]	negative character class - matches any character not specified

- [flags]

  CondPattern에 [flags]를 사용하여 패턴 매칭 방식을 변경할 수 있다.

지시자	설명
nocase|NC	패턴을 매칭할 때 대소문자를 구분하지 않는다.
ornext|OR	RewriteCond 다음에 또 다른 RewriteCond가 있을 경우 다음 RewriteCond와 logical OR로 조합되도록 설정한다. 명시적으로 설정하지 않았을 경우는 항상 AND로 다음 조건과 조합한다.

지시자	설명
novary|NV	HTTP Header를 TestString으로 사용할 경우 Vary Header가 Response에 추가되지 않도록 한다.

다음은 [OR] flag 사용한 예이다.

```
RewriteCond %{HTTP_HOST}  ^host1.*  [OR]
RewriteCond %{HTTP_HOST}  ^host2.*  [OR]
RewriteCond %{HTTP_HOST}  ^host3.*
RewriteRule ...
```

[OR] flag를 사용하지 않을 경우 3개의 RewriteCond/RewriteRule을 각각 생성해야 한다.

4.7.4. RewriteRule

RewriteRule은 rewriting 동작을 결정하며, 다음과 같은 형식으로 설정한다. 사용자 요청이 RewriteCond에 매칭되는 경우 RewriteRule 설정에 의해 사용자 요청 중 Pattern을 Substitution으로 교체한다.

```
RewriteRule <Pattern> <Substitution> [flags]
```

- <Pattern>
  - Pattern은 Perl compatible requlat expression을 사용할 수 있으며, '!'을 사용하면 패턴에 매칭되지 않는 경우를 지정할 수 있다.

    '!'을 사용할 경우 괄호로 묶은 그룹 패턴($N)을 사용할 수 없다. 패턴이 매칭되지 않는 경우이므로, 그룹 패턴($N)에 값이 없기 때문이다.

  - RewriteRule만 단독으로 설정할 경우 패턴 매칭은 URL-path를 사용하며, RewriteCond와 함께 사용될 경우 마지막에 매칭된 패턴을 사용한다.
- <Substitution>

  URL을 어떤것으로 교체할지 설정하며, 다음과 같은 값을 사용할 수 있다.

설정값	설명
file-system path	'/'로 시작하는 file-system의 절대 경로를 설정할 경우 해당 파일을 사용자 응답에 사용한다. 단, 설정한 경로가 file-system에 존재해야 한다.
URL-path	일반적인 URL-path를 설정할 경우 해당 리소스를 사용한다.
Absolute URL	"http://<hostname>/file.html"처럼 Absolute URL을 사용할 경우에는 hostname이 서버와 일치하면 scheme과 hostname을 제외한 나머지를 URL-path처럼 사용하고, 일치하지 않을 경우 외부 서버로 redirect하도록 처리한다.
-	'-'는 교체하지 않는다는 의미이다.
$N (N=0..9)	RewriteRule의 패턴 중 N번째 그룹 패턴을 지칭한다.
%N (N=1..9)	마지막에 매칭된 RewriteCond의 패턴 중 N번째 그룹 패턴을 지칭한다.

설정값	설명
%{VARNAME}	서버에서 제공하는 VARNAME을 참조한다. RewriteCond의 TestString에 적용되는 항목을 사용할 수 있다.
?	기본적으로 Query string은 변경되지 않지만, 이를 교체하고 싶을 경우 '?'를 Substitution안에 추가한다. Query string을 삭제하고자 하는 경우 Substitution의 마지막이 '?'로 끝나도록 한다.

Apache의 mod_rewrite에서 제공하는 RewriteMap을 지원하지 않기 때문에 '${mapname:key|default}'는 사용할 수 없다.

- [flags]
  - RewriteRule의 세부 동작을 [flags]의 설정을 통해 조절할 수 있다.
  - flags는 콤마(,)를 사용하여 여러 개를 설정할 수 있으며, 다음과 같은 값을 사용할 수 있다.
    - B (escape backreferences)

      Substitution에 사용되는 backreference($N 혹은 %N)는 URL의 %-encoding을 unescape하여 사용한다. [B] 옵션을 사용하면, URL의 %-encoding을 그대로 사용할 수 있다.

      예를 들어, 다음과 같이 설정했을 경우 "/C%2b%2b"가 unescape되어 "index.php?show=/C++"로 매핑되지만, [B] 옵션을 사용했을 경우에는 "index.php?show=/C%2b%2b"로 매핑할 수 있다.

      ```
      RewriteRule ^(.*)$ index.php?show=$1
      ```

    - chain|C (chained with next rule)

      이 옵션을 설정했을 경우 rule이 매칭되지 않았을 때 다음 rule은 체크하지 않게 된다. 그 다음 rule또한 [C] 옵션을 사용했다면, 마찬가지로 건너뛰게 된다.

    - cookie|CO=NAME:VAL:domain[:lifetime[:path[:secure[:httponly]]]] (set cookie)

      응답에 Set-Cookie Header를 추가하여 사용자 브라우저에 Cookie를 추가한다.

    - discardpathinfo|DPI (discard PATH_INFO)

      디렉터리별 context를 구성한 환경에서 RewriteRule은 URI와 PATH_INFO를 구별하여 매번 rule을 적용할 때 마다 URI와 PATH_INFO를 결합하게 되는데, URI가 여러 번 매칭되는 경우 PATH_INFO가 여러 번 붙게 된다.

      [DPI] 옵션은 PATH_INFO를 구분하지 않고 RewriteRule을 적용하도록 한다.

    - env|E=VAR:VAL (set environment variable)

      환경변수에 VAR=VAL를 추가한다. VAL에는 regexp backreferences($N 혹은 %N)를 사용할 수 있다. 이 환경변수는 SSI나 CGI에서 사용할 수 있으며, RewriteCond의 패턴 중 %{ENV:VAR}으로 사용할 수 있다.

    - forbidden|F (force URL to be forbidden)

"403 Forbidden" 응답을 보낸다.

- gone|G (force URL to be gone)

  "410 Gone" 응답을 보낸다.

- handle|H=Content-handler (force Content handler)

  Content-handler를 설정한다.

- last|L (last rule)

  rewriting 과정을 이곳에서 종료하도록 한다. C의 break 명령과 유사한 기능을 하는 옵션이다.

- next|N (next round)

  지금까지 변경된 URL을 가지고 처음부터 다시 rewriting 과정을 수행하도록 한다. C의 continue 명령과 유사한 기능을 하는 옵션이다. 무한루프가 발생할 수 있으니 주의해야 한다.

- nocase|NC (no case)

  패턴에 대소문자를 구분하지 않도록 한다. [A-Z]와 [a-z]를 똑같이 취급한다.

- noescape|NE (no URI escaping of output)

  rewriting 과정에서 URL을 %-encoding하지 않도록 한다.

- nosubreq|NS (not for internal sub-requests)

  내부 요청일 경우 rewriting을 중지하도록 한다.

- passthrough|PT (pass through to next handler)

  rewriting을 수행한 결과를 다른 handler가 사용할 수 있도록 한다.

- qsappend|QSA (query string append)

  query string을 rewriting하는 경우 기존의 query string을 덮어쓰는 대신, 이를 그대로 유지하면서 새로운 query string을 추가하고자 하는 경우 이 옵션을 설정한다.

- redirect|R[=code] (force redirect)

  Substitution이 Absolute URL일 경우 hostname이 서버의 호스트와 일치하는 경우에도 강제로 다시 redirect하도록 한다.

  code를 지정하지 않았을 경우 302 Moved Temporarily가 사용되며, code에는 Status Code를 직접 입력하거나, temp(default), permanent, seeother를 설정할 수 있다.

- skip|S=num (skip next rules)

  다음 num개의 rule를 건너뛰도록 한다.

- type|T=MIME-type (force MIME-Type)

응답의 Content-Type을 설정한다.

4.7.5. 예제

URL Rewrite의 예제는 다음과 같다.

- 예제 1

  다음은 "www.test.com/"과 같은 URL 패턴을 매칭하여, "www.test.com/rewrite.html"로 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex1
RewriteCond %{HTTP_HOST} ^www\.test\.com$       # if {Host} == "www.test.com"
RewriteRule ^/$ /rewrite.html [L]               # then "/" > "/rewrite.html"
```

- 예제 2

  다음은 "www.test.com/temp/xxx.html"과 같은 요청을 할 때 temp 디렉터리에 xxx.html 파일이 없는 경우 "www.test.com/temp/temp_error.html"로 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex2
RewriteCond %{REQUEST_FILENAME} !-f
# if {요청파일명} != 파일을 가리키거나 포함
RewriteRule ^/([^/]+) /$1/$1_error.html [L]
# then "/temp/xxx.html" > "/temp/temp_error.html"
```

- 예제 3

  다음은 "http://www.test.com:80"과 같은 요청을 할 때 "https://www.test.com:443"으로(http 요청을 https 요청으로) 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex3
RewriteCond %{HTTP_HOST} ^www\.test\.com$
# if {Host} == "www.test.com"
RewriteCond %{SERVER_PORT} 80
# AND {Port} == "80"
RewriteRule .* https://www.test.com:443$0 [R]
# then > "https://www.test.com:443$0"   ($0: request uri)
```

- 예제 4

  다음은 "www.test.com/xxx.html"과 같은 URL 패턴을 매칭하여 Request URI는 무시하고 무조건 "www.test_new.com/"으로 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex4
RewriteCond %{HTTP_HOST} ^www\.test\.com$       # if {Host} == "www.test.com"
RewriteRule .* http://www.test_new.com [R]       # then > "http://www.test_new.com"
```

- 예제 5

다음은 "www.test.com/test.html"과 같은 URL 패턴을 매칭하여 "www.test_new.com/test.html"로 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex5
RewriteCond %{HTTP_HOST} ^www\.test\.com$
# if {Host} == "www.test.com"
RewriteRule .* http://www.test_new.com$0 [R]
# then > "http://www.test_new.com$0"   ($0: request uri)
```

- 예제 6

다음은 Request Method가 POST이고 Referer Header가 없을 경우 "403 Forbidden" 에러로 처리하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex6
RewriteCond %{REQUEST_METHOD} POST              # if {Method} == "POST"
RewriteCond %{HTTP_REFERER} =""                 # AND {Referer} == ""
RewriteRule . - [F]                             # then > 403 Forbidden Return
```

- 예제 7

다음은 "/../" 으로 시작하는 요청의 경우 "/403.html"로 처리하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex7
RewriteRule ^/\.\./ /403.html [L]               # "/../"로 시작 > "/403.html"
```

- 예제 8

다음은 "aaa.test.com/xxx.html"로 요청할 경우 "www.test.com/aaa/xxx.html"로, "bbb.test.com/xxx.html"로 요청할 경우 "www.test.com/bbb/xxx.html"로 처리하는

```
# url rewrite config - ex8
RewriteCond %{HTTP_HOST} ^(aaa|bbb)\.test\.com
# if {Host} == ("aaa.test.com" OR "bbb.test.com")
RewriteRule .* /%1$0 [L]
# then "/xxx.html" > "/(aaa|bbb)/xxx.html" (%1: RewriteCond의 첫 번째 괄호, $0: request uri)
```

- 예제 9

다음은 "www.test.com/test.do?query=value1"로 요청할 경우 "www.test.com:8080/test.do?query=value1"로, "www.test.com/test.do?query=value2&.."으로 요청할 경우 "www.test.com:8080/test.do?query=value2&.."로 Query string에 따라 포트 변경 처리를 하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex9
RewriteCond %{QUERY_STRING} ^query=value1$ [OR]
# if {QueryString} == "query=value1"
RewriteCond %{QUERY_STRING} ^query=value2&
# OR {QueryString} == "query=value2&.."
RewriteRule .* http://www.test.com:8080$0 [R]
```

```
# then > "http://www.test.com:8080$0"   ($0: request uri)
```

- 예제 10

  다음은 "aaa.test.com/test.html"로(www 로 시작하지 않는) 요청할 경우에 "www.test.com/aaa/test.html"로 처리하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex10
RewriteCond %{HTTP_HOST} !^www\.test\.com$
# if {Host} != www.test.com
RewriteCond %{HTTP_HOST} ^([a-zA-Z0-9]+)\.test.com$
# AND {Host} == "(영문숫자조합).test.com"
RewriteRule  .* /%1/$0 [L]
# then > "(영문숫자조합)/$0"   (%1: RewriteCond의 첫 번째 괄호, $0: request uri)
```

- 예제 11

  다음은 Request Header에 Referer Header가 없이 css나 js 파일을 요청하는 경우 403 Forbidden으로 처리하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex11
RewriteCond %{HTTP_REFERER} !.        # if {HTTP_REFERER} == ""
RewriteRule \.(css|js)$ - [F]         # then "*.css|*.js" > 403 Forbidden Return
```

- 예제 12(REQEUST_URI)

  다음은 "/user@somehost.com/"과 같은 URL 패턴을 매칭하여 "/req_test.jsp?blogId=user@somehost.com"으로 변환하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex12 (REQEUST_URI)
RewriteCond %{REQUEST_URI} /([a-zA-Z0-9_-]+(@([a-zA-Z0-9_-]+).(com|net|co.kr))?)/?$
RewriteRule . /req_test.jsp?blogId=%1 [PT,L]
```

- 예제 13(HTTP_HOST)

  다음은 HTTP Request Header의 호스트가 "tmaxsoft.com"으로 끝나고, URL이 "/redirect/"로 시작하는 경우를 매칭하여 "http://www.tmaxsoft.com/redirect.html"로 redirect하도록 하는 URLRewriteConfig 설정 예제이다.

```
# url rewrite config - ex13 (HTTP_HOST)
RewriteCond %{HTTP_HOST} tmaxsoft.com$
RewriteCond %{REQUEST_URI} /redirect/.*$
RewriteRule . http://www.tmaxsoft.com/redirect.html [R,L]
```

4.8. Filter 모듈 개발(Enterprise+)

WebtoB를 설치할 때 제공되는 usrinc/httpfilt.h 파일은 ISAPI(Internet Server Application Program Interface) Filter Interface를 제공한다. 해당 파일을 include하여 Filter 모듈을 개발할 수 있다. WebtoB

Enterprise에서 제공되는 Filters process를 사용할 수 있다.

다음은 클라이언트 및 요청 정보 등을 확인하는 filter sample source 예제이다.

```
/**
 * file: sample.c for sample_filter
 * Linux 32bit에서 compile 예
 *   $ cc -m32 -D_REENTRANT -fPIC -I/webtob/usrinc -g -pthread -o sample.so sample.c -shared
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "httpfilt.h"


#define REDIRECT_PAGE "302 Moved Temporarily"

int wb_handler; /* WebtoB HTH count for only filters process */
time_t current_time;

int init()
{
  int n;

  /* set current time */
  time(&current_time);

  n = 0;
  /* n = init_check(); */

  return n;
}

void url_redirect(HTTP_FILTER_CONTEXT *pfc, char *url)
{
  pfc->ServerSupportFunction(pfc,
                             SF_REQ_SEND_RESPONSE_HEADER,
                             (LPVOID) REDIRECT_PAGE,
                             (LPDWORD) url,
                             0);
  return;
}

/**
 * called when WebtoB is booted..
**/
BOOL WINAPI
GetFilterVersion(HTTP_FILTER_VERSION * pVer)
{
  DWORD dwRet;

  /* Version 4.0 */
  /* pVer->dwFilterVersion = MAKELONG(0, 4); */
  pVer->dwFilterVersion = HTTP_FILTER_REVISION;
  strncpy(pVer->lpszFilterDesc, "filter sample", SF_MAX_FILTER_DESC_LEN);

  /* init when webtob is booting.. */
```

```
  wb_handler = 1; /* default */
#if 0
  /* only filters process is used.. */
  if (0 < pVer->dwFlags) {
    wb_handler = pVer->dwFlags;
    pVer->dwFlags = 0;
    dwRet = init();
    if (dwRet < 0) {
      /* init failed.. */
      return 0;
    }
  }
#endif

  /* flags */
  pVer->dwFlags = (
        SF_NOTIFY_SECURE_PORT        |
        SF_NOTIFY_NONSECURE_PORT     |
        SF_NOTIFY_READ_RAW_DATA      |
        SF_NOTIFY_PREPROC_HEADERS    |
        SF_NOTIFY_URL_MAP            |
        SF_NOTIFY_AUTHENTICATION     |
        SF_NOTIFY_SEND_RAW_DATA      |
        SF_NOTIFY_LOG                |
        SF_NOTIFY_END_OF_NET_SESSION |
        SF_NOTIFY_ORDER_DEFAULT
        );

  return 1;
}


/**
 * run-time callback
**/
DWORD WINAPI HttpFilterProc(HTTP_FILTER_CONTEXT *pfc, DWORD NotificationType, VOID * pvNotification)
{
  DWORD dwRet;

  /* init when the first request.. */
  dwRet = init();
  if (dwRet < 0) {
    /* init failed.. */
    char redirect_page[256];

    sprintf(redirect_page, "Location: %s\r\n", "error.html");
    url_redirect(pfc, redirect_page);

    return SF_STATUS_REQ_FINISHED;
  }

  printf("\n\n start -> HttpFilterProc: NotificationType=%d, hth=%d \n",
         NotificationType, pfc->ulReserved);

  /* Direct the notification to the appropriate routine for processing. */
  switch (NotificationType) {
    case SF_NOTIFY_PREPROC_HEADERS:
      dwRet = process_preproc_headers(pfc, (PHTTP_FILTER_PREPROC_HEADERS) pvNotification);
      break;
```

```
    case SF_NOTIFY_SEND_RAW_DATA:
      dwRet = SF_STATUS_REQ_FINISHED_KEEP_CONN;
      break;
    case SF_NOTIFY_READ_RAW_DATA:
    case SF_NOTIFY_URL_MAP:
    case SF_NOTIFY_AUTHENTICATION:
    case SF_NOTIFY_LOG:
    case SF_NOTIFY_END_OF_NET_SESSION:
    default:
      dwRet = SF_STATUS_REQ_NEXT_NOTIFICATION;
      break;
  }
  return dwRet;
}

/**
 * process_preproc_headers
 *  GetHeader()
 *   - get the request header information
 *  GetServerVariable()
 *   - get the server variable information
 *  AddResponseHeaders()
 *   - add Response Header
 *  SetHeader()
 *   - add Request Header
 *  ServerSupportFunction()
 *   - set 302 Redirect page
**/
DWORD process_preproc_headers(HTTP_FILTER_CONTEXT *pfc, HTTP_FILTER_PREPROC_HEADERS *pvNotification)
{
  /* watch out buffer size */
  int size;
  char request_method[32];
  char request_uri[8192];
  char client_ip[256];
  char server_port[32];
  char content_type[256];
  char content_length[256];
  char body[10240];
  char s_header[8192];
  char cookie[8192];
  char add_res_header[8192];
  char redirect_page[8192];

  int redirect_test = 0;
  int setcookie_test = 1;

  /* ssl y/n */
  printf("Request is SSL? %d \n", pfc->fIsSecurePort);

  /* get request method */
  size = sizeof(request_method);
  memset(request_method, 0, size);
  /* pvNotification->GetHeader(pfc, "method", request_method, &size); */
  pfc->GetServerVariable(pfc, "REQUEST_METHOD", request_method, &size);
  printf("Request Method? %s \n", request_method);

  /* get request url */
  size = sizeof(request_uri);
```

```
memset(request_uri, 0, size);
/* pvNotification->GetHeader(pfc, "url", request_uri, &size); */
pfc->GetServerVariable(pfc, "REQUEST_URI", request_uri, &size);
printf("Request url? %s \n", request_uri);

/* get client ip */
size = (int) sizeof(client_ip);
memset(client_ip, 0, size);
pfc->GetServerVariable(pfc, "REMOTE_ADDR", client_ip, &size);
printf("Client ip? %s \n", client_ip);

/* get server port */
size = (int) sizeof(server_port);
memset(server_port, 0, size);
pfc->GetServerVariable(pfc, "SERVER_PORT", server_port, &size);
printf("Server Port? %s \n", server_port);

/* get Content-Type Request Header */
size = (int) sizeof(content_type);
memset(content_type, 0, size);
pvNotification->GetHeader(pfc, "Content-Type", content_type, &size);
printf("Content-Type? %s \n", content_type);

/* get Content-Length Request Header */
size = (int) sizeof(content_length);
memset(content_length, 0, size);
pvNotification->GetHeader(pfc, "Content-Length", content_length, &size);
printf("Content-Length? %s \n", content_length);

/* get Cookie Request Header */
size = (int) sizeof(cookie);
memset(cookie, 0, size);
pvNotification->GetHeader(pfc, "Cookie", cookie, &size);
printf("Cookie? %s \n", cookie);

/* get special header */
size = sizeof(s_header);
memset(s_header, 0, size);
pvNotification->GetHeader(pfc, "Special-Header", s_header, &size);
printf("Special-Header? %s \n", s_header);

/* get body data */
if (content_length) {
  size = (int) sizeof(body);
  memset(body, 0, size);
  pvNotification->GetHeader(pfc, "body", body, &size);
  printf("Body? %s \n", body);
}

/* add Response Header on processing request */
if (setcookie_test) {
  sprintf(add_res_header, "Set-Cookie: %s=%s; path=/; domain=test\r\n", "TestKey", "TestValue");
  pfc->AddResponseHeaders(pfc, add_res_header, 0);
}
sprintf(add_res_header, "AddResponseHeader: test");
pfc->AddResponseHeaders(pfc, add_res_header, 0);

/* add Request Header */
pvNotification->SetHeader(pfc, "addReuestHeader", "test value");
```

```
  /* 302 redirect */
  if (redirect_test) {
    sprintf(redirect_page, "Location: %s\r\n", "test/redirect.html");
    url_redirect(pfc, redirect_page);

    return SF_STATUS_REQ_FINISHED;
  }

  return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

5. 기동 및 종료

본 장에서는 환경설정을 마친 WebtoB를 기동 및 종료하기 위한 명령어와 각종 옵션에 대해 설명한다.

5.1. WebtoB 기동

5.1.1. 기동 전 점검할 사항

WebtoB를 기동하기 전, 다음과 같은 사항을 점검해야 한다.

- 이진 WebtoB 환경설정 파일이 존재하는가 ?
- NODE 절의 WebtoBDir 항목에 설정된 경로에 WebtoB 실행 프로그램들(wsm, htl, hth, htmls, cgis, ssis 등)이 존재하는가 ?

위의 사전점검 항목이 확인되었다면 WebtoB 시스템을 기동할 준비가 된 것이다.

5.1.2. wsboot

WebtoB 시스템을 기동하기 위해서 wsboot라는 프로그램이 제공된다.

- 사용법

```
> wsboot [-f 이진 WebtoB 환경 파일 이름][-T][-i][-c 설정 파일 이름][-g 서버 그룹 이름]
       [-A][-S 서버 이름][-s 서버 이름][-k number 개수][-w][-b]
       [-B Block Listen 상태로 boot][-o CLOPT string][-h]
```

  - WebtoB를 기동할 때 사용하는 옵션

옵션	설명
[-f 이진 *WebtoB* 환경 파일 이름]	WebtoB는 환경 파일을 토대로 기동된다. 옵션 입력 항목에 참조할 이진 WebtoB 환경 파일명을 설정한다. 파일명을 지정하지 않으면 기본값으로 wsconfig 파일이 참조된다.
[-T]	WebtoB 관리 프로세스들(WSM, HTL, HTH)만을 기동시킨다. 실제 서비스를 하기 위해서는 각 서버들을 기동시켜야 한다.
[-i]	"wscfl -i http.m"을 수행한 후 wscfl이 성공하면 WebtoB를 기동한다.
[-c 설정 파일 이름]	"wscfl -i 설정 파일 이름"을 수행한 후 wscfl이 성공하면 WebtoB를 기동한다.

  - wsboot 후 적용되는 옵션(프로세스를 기동할 때 사용)

옵션	설명
[-g 서버 그룹 이름]	지정된 서버 그룹에 존재하는 서버 프로세스들을 기동시킨다. 서버 그룹명은 WebtoB 환경 파일 내의 SRVGROUP 절에 미리 정의되어 있어야 한다.
[-A]	WebtoB 환경 파일 내의 SERVER 절에 정의된 모든 서버 프로세스들을 기동시킨다. 이 옵션을 사용하기 위해서는 [-T] 옵션 등을 사용하여 WebtoB 관리 프로세스들이 기동되어 있는 상태여야 한다.
[-S 서버 이름]	지정된 서버에 존재하는 서버 프로세스들을 Min 개수만큼 기동시킨다.
[-s 서버 이름]	지정된 서버 프로세스만을 기동시킨다. 서버 프로세스 이름은 WebtoB 환경 파일 내의 SERVER 절에 미리 정의되어 있어야 한다. **[-k]** 옵션을 함께 사용하여 서버 프로세스 개수를 지정할 수 있다. 서버 프로세스 개수는 현재 기동되어 있는 개수를 포함하여 SERVER 절의 Max 항목에 정의된 개수를 넘어서는 안된다. [-k] 옵션을 생략하면 서버 프로세스는 1개만 기동된다.
[-k *number* 개수]	[-s] 옵션과 함께 사용해야 한다.
[-w]	WebtoB 서버 프로세스들을 한 번에 기동시키는 것이 아니라 한 번에 하나씩 기동시킨다.
[-b]	백업 서버가 지정되어 있는경우 백업 서버도 함께 기동시킨다.
[-B *Block Listen* 상태로 *boot*]	클라이언트의 요청을 받아들이지 않는 상태로 WebtoB를 기동시킨다.
[-o CLOPT string]	[-o] 파일명, **standard output**을 저장한다.
[-h]	도움말을 보여준다.

- 예제

  다음은 wsconfig 환경 파일을 참조하여 WebtoB 관리 프로세스와 서비스 서버 프로세스들을 모두 기동시키는 예제이다.

```
$ wsboot
```

  다음은 wsconfig2 환경 파일을 참조하여 WebtoB 관리 프로세스인 WSM, HTL, HTH만을 기동시키는 예제이다.

```
$ wsboot –T –f wsconfig2
```

  다음은 wsconfig 환경 파일을 참조하여 모든 서비스의 서버 프로세스들을 기동시키는 예제이다.

```
$ wsboot -A
```

  다음은 환경 파일을 참조하여 svr1 프로세스를 MIN 개수만큼 기동시키는 예제이다.

```
$ wsboot –S svr1
```

다음은 wsconfig2 환경 파일을 참조하여 svr1 프로세스를 5개 기동시키는 예제이다.

```
$ wsboot -s svr1 –k 5 -f wsconfig2
```

WebtoB 시스템을 기동시키면 먼저 이진 WebtoB 환경 파일 내용이 공유 메모리에 적재되고(Load) 적재된 환경 파일 노드마다 지정된 WEBTOBDIR 디렉터리에서 WebtoB 기능 프로세스(WSM, HTL, HTH)들이 기동된다. 그 후 등록된 서비스의 서버 프로세스들이 차례로 기동된다.

WebtoB 프로세스들의 기동 순서는 다음과 같다.

```
WSM → HTL → HTH → 서버 프로세스들
```

5.2. WebtoB 종료

WebtoB 시스템 종료도 wsboot와 마찬가지로 이진 WebtoB 환경 파일로 이루어진다. 시스템에서 사용하던 공유 메모리를 없애고 기동된 WebtoB 프로세스들(WSM, HTL, HTH)과 응용 서버 프로세스들을 종료시킨다.

5.2.1. wsdown

WebtoB 시스템을 종료시키기 위해서는 wsdown이라는 명령을 사용한다. 사용되는 옵션들은 wsboot 명령과 유사하다.

- 사용법

```
wsdown [-f 이진 WebtoB 환경 파일 이름][-A]
       [-S 서버 이름][-s 서버 이름][–k 개수]
       [-g 서버 그룹 이름][-p 서버 번호]
       [-i][-w 지연시간(5초 이하)][-k 개수][-h]
```

옵션	설명
[-f 이진 *WebtoB* 환경 파일 이름]	참조할 이진 WebtoB 환경 파일명을 지정해야 한다. 파일명을 지정하지 않으면 기본값으로 wsconfig 파일이 참조된다.
[-A]	모든 서비스 서버 프로세스들을 종료시킨다.
[-S 서버 이름]	현재 사용 중인 해당 서비스 서버 프로세스들을 종료시킨다.
[-s 서버 이름]	해당 서버 프로세스만을 종료시킨다. wsboot와 마찬가지로 지정된 개수만큼 동작 중인 해당 서버 프로세스를 종료시킨다. [-k] 옵션을 생략하면 서버 프로세스는 1개만 종료된다.
[–k 개수]	[–s] 옵션과 함께 사용해야 한다.

옵션	설명
[-g 서버 그룹 이름]	해당 서버 그룹의 서버 프로세스들을 종료시킨다. 사용되는 그룹 이름은 WebtoB 환경 파일의 SVRGROUP 절에 정의된 것이어야 한다.
[-p 서버 번호]	특정 서버 프로세스를 종료시킨다. **[-s]** 옵션에 의한 종료와 달리 wsadmin에서 “st –p” 명령으로 확인할 수 있는 프로세스 번호(spr_no)를 사용하여 특정 프로세스를 종료시킬 수 있다.
[-i]	wsdown 명령을 즉시 수행한다. 기본적으로 wsdown 명령은 해당 업무를 모두 종료하고 수행되지만 **[-i]**(Immediately) 옵션에 의한 종료는 현재 수행 중인 업무를 무조건 중단하기 때문에 신중하게 사용해야 한다.
[-w 지연시간*(5*초 이하*)*]	wsdown 명령을 지정한 시간 후에 실행한다. 시간 지정은 5초 이하로 지정해야 한다.
[-h]	도움말을 보여준다.

6. WebtoB 관리

WebtoB는 시스템 동작 중 각종 상태 정보 확인, 제어 등의 관리기능을 제공한다. 예를 들어 WebtoB 시스템의 종료 및 기동의 절차를 거치지 않고도 현재 환경설정에 대한 정보를 확인해 보고 이를 동적으로 변경할 수 있다. 또한 제공되는 서비스들 중 특정 서비스에 대하여 현재 처리 상태 즉, 몇 건의 서비스를 처리해 왔으며, 평균 처리시간이 얼마나 되며, 몇 건의 요청이 대기하고 있는지 등의 정보를 확인할 수 있다.

WebtoB는 이와 같은 기능들을 콘솔 관리자 프로그램 wsadmin과 브라우저를 위한 WebAdmin을 통해서 제공한다. 본 장에서는 WebtoB 관리를 위해 제공하는 프로그램과 명령어에 대해서 설명한다.

6.1. wsadmin 콘솔 관리자 프로그램

wsadmin은 텍스트 기반의 관리환경을 제공한다. 항상 프롬프트(prompt) 상태로 대기 중이다가 입력되는 명령어를 해석하여 이를 실행하게 된다.

wsadmin 프로그램은 다음과 같이 실행한다.

```
$ wsadmin
```

이후 아래와 같은 메시지와 함께 프롬프트(prompt)가 나타난다. 이것이 wsadmin 프로그램 실행 상태임을 나타낸다.

```
--- Welcome to WebtoB Admin (Type "quit" to leave) ---
$$1 webtob (wsadm) [2016-02-03T15:07:52]:
```

wsadmin 툴에서 사용할 수 있는 명령어들의 도움말을 조회할 때는 **'help <명령어>**'가 사용된다.

```
$$2 webtob (wsadm) [2016-02-03T15:08:07]: help st
Summary: stat, st: thread(process) and service state statistics
Usage: stat (st) -v [servername] | -j [jsvservername] | -p [sprname] |
                 -s [servicename] | -rpg [rpgname] | -rp [rproxyname] |
                 -h [hthno] | -t | - T
                 -v [servername1,servername2,..] : status for each Server
                 -j [jsvservername1, jsvservername2,..] : status for each JSV Server(Jengine)
                 -p [sprname1,sprname2,..] : status for each Server Process
                 -s [servicename1,servicename2,..] : status for each Service
                 -rpg [rpgname1,rpgname2,..] : status for each ReverseProxyGroup
                 -rp [rproxyname1,rproxyname2,..] : status for each ReverseProxy
                 -h [hthno] : statistics of each HTH
                 -t : statistics of each HTH threads
                 -T : statistics of HTH threads group
```

wsadmin 툴을 종료하기 위해서는 quit (q) 명령을 사용한다.

```
$$3 webtob (wsadm) [2016-02-03T15:10:16]: quit
```

다음은 wsadmin이 제공하는 명령어들이다.

명령어	약자	설명
cacherefresh	(cr)	HTTP 응답 캐시에 저장된 응답을 삭제한다.
cachelist		HTTP 응답 캐시에 저장된 응답들의 정보를 출력한다.
cliinfo	(ci)	접속 웹 브라우저를 확인한다.
clilisten	(cl)	WebtoB의 클라이언트의 Listen Port를 제어한다.
config	(cfg)	환경설정 내용을 조회한다.
discon	(ds)	접속 중인 클라이언트 연결을 강제로 해제한다.
history	(hist)	최근 수행한 50개 명령을 조회한다.
help	(h)	도움말을 조회한다.
hthmem		HTH의 메모리 사용 내역을 파일에 기록한다.
ll		로그 레벨을 동적으로 변경한다.
logend	(loge)	logging을 종료한다.
logstart	(logs)	logging을 시작한다.
logsearch		로그 파일의 내용을 검색한다.
patchinfo		릴리즈된 이후 패치된 정보를 조회한다.
qpurge	(qp)	큐에 적체된 요청을 삭제한다.
quit	(q)	wsadmin을 종료한다.
rebootsvr	(rbs)	서버 프로그램을 교체한다.
repeat	(r)	명령어를 반복한다.
restat		서버 프로세스의 통계 정보를 초기화한다.
resume	(rs)	중지된 서버 프로세스를 재개한다.
set		현재 설정된 환경값을 동적으로 변경한다.
stat	(st)	프로세스, 스레드 및 서비스 상태에 대한 통계를 조회한다.
suspend	(sp)	동작 중인 서버 프로세스를 중지한다.
svrinfo	(si)	서버 정보를 확인한다.
webtobinfo	(wi)	WebtoB 시스템의 정보를 확인한다.
wsboot	(boot)	WebtoB를 시작한다. wsboot와 동일하다.
wsdown	(down)	WebtoB를 종료한다. wsdown과 동일하다.
!		직전 명령어를 반복한다.

6.1.1. 환경정보

6.1.1.1. webtobinfo (wi)

WebtoB 시스템의 환경정보를 조회한다. 버전, 최대 사용자수, 사용 만료기한 정보 등을 확인할 수 있다.

- 사용법

```
> wi
```

- 예제

```
$$1 webtob (wsadm) [2016-02-03T15:12:27]: wi

License: CLOUD Enterprise edition
Version=WebtoB 5.0 SP 0 Fix #0 Linux-K2.6_x86 FD16384 B41 epoll 2016/02/03

         maxuser = UNLIMITED,
         node_count = 1,
         svgrpcount = 0,
         svr_count = 0, svc_alloc_count = 512

WebtoB All Node Info: node_count = 1:
-------------------------------------------------------------------------
  no   name     nodeport  racport  shmkey  shmsize0 shmsize1 shmsize2 hth
-------------------------------------------------------------------------
   0   webtob     7777     3333    196608    200360  1978068     1748   1
```

6.1.1.2. config (cfg)

현재 동작 중인 시스템의 환경정보를 조회한다. 즉, 환경 파일에서 정의된 도메인, 노드, 서버 그룹, 서버, 서비스별로 기본값까지 포함한 모든 환경정보를 확인할 수 있다.

- 사용법

```
> config [-d][-n][-vh VHOST 이름][-g SVRGROUP 이름][-v SERVER 이름]
       [-s SERVICE 이름][-dir DIRECTORY 이름][-u URI 이름][-a ALIAS 이름]
       [-l LOGGING 이름][-e EXT 이름][-ssl SSL 이름][-pssl PROXY_SSL 이름]
       [-tcpgw TCPGW 이름][-rproxy REVERSE_PROXY 이름]
       [-rpg REVERSE_PROXY_GROUP 이름][-ll LOGLEVEL 이름][-headers HEADERS 이름]
       [-access ACCESS 이름][-pc PRECEDING_COMMAND 이름]
       [-t HTH_THREAD 이름]
```

옵션	설명
[-d]	DOMAIN 절을 설정한다.
[-n]	NODE 절을 설정한다.
[-vh *VHOST* 이름]	VHOST 절 전체 또는 지정한 이름의 VHOST를 설정한다.
[-g *SVRGROUP* 이름]	SVRGROUP 절 전체 또는 지정한 이름의 서버 그룹을 설정한다.
[-v *SERVER* 이름]	SERVER 절 전체 또는 지정한 이름의 서버를 설정한다.

옵션	설명
[-s *SERVICE* 이름]	SERVICE 절 전체 또는 지정한 이름의 서비스를 설정한다.
[-dir *DIRECTORY* 이름]	DIRECTORY 절 전체 또는 지정한 이름의 디렉터리를 설정한다.
[-u *URI* 이름]	URI 절 전체 또는 지정한 이름의 URI를 설정한다.
[-a *ALIAS* 이름]	ALIAS 절 전체 또는 지정한 이름의 ALIAS를 설정한다.
[-l *LOGGING* 이름]	LOGGING 절 전체 또는 지정한 이름의 LOGGING을 설정한다.
[-e *EXT* 이름]	EXT 절 전체 또는 지정한 이름의 EXT를 설정한다.
[-ssl *SSL* 이름]	SSL 절 전체 또는 지정한 이름의 SSL을 설정한다.
[-pssl *PROXY_SSL* 이름]	PROXY_SSL 절 전체 또는 지정한 이름의 PROXY_SSL을 설정한다.
[-tcpgw *TCPGW* 이름]	TCPGW 절 전체 또는 지정한 이름의 TCP 게이트웨이를 설정한다.
[-rproxy *REVERSE_PROXY* 이름]	REVERSE_PROXY 절 전체 또는 지정한 이름의 REVERSE_PROXY를 설정한다.
[-rpg *REVERSE_PROXY_GROUP* 이름]	REVERSE_PROXY_GROUP 절 전체 또는 지정한 이름의 REVERSE_PROXY_GROUP을 설정한다.
[-ll *LOGLEVEL* 이름]	LOGLEVEL 절 전체 또는 지정한 이름의 LOGLEVEL을 설정한다.
[-headers *HEADERS* 이름]	HEADERS 절 전체 또는 지정한 이름의 HEADERS를 설정한다.
[-access *ACCESS* 이름]	ACCESS 절 전체 또는 지정한 이름의 ACCESS를 설정한다.
[-pc *PRECEDING_COMMAND* 이름]	PRECEDING_COMMAND 절 전체 또는 지정한 이름의 PRECEDING_COMMAND를 설정한다.
[-t *HTH_THREAD* 이름]	HTH_THREAD 절 전체 또는 지정한 이름의 HTH_THREAD를 설정한다.

- 예제

  다음은 NODE 절 환경설정을 출력한 예제이다. NODE 절의 설정 항목에 대한 자세한 설명은 설정 항목을 참고한다.

```
$$9 webtob (wsadm) [2016-02-03T15:14:46]: cfg -n
    NODE(0): Name = webtob,
         HostName = "webtob",
         DocRoot = "/root/webtob_docroot/",
         SvrRoot = "/root/wb-5000-clean/",
         Method = "GET, POST, HEAD, OPTIONS",
         ShmKey = 54000,
         Hth = 1,
         HthQTimeout(hqt) = 0,
         NodePort = 7777,
         Port = "8080",
         JsvPort = 9999,
         ...
         Logging = "log1",
         ErrorLog = "log2",
         SysLog = "syslog",
         ...
         CheckURL = Y,
         CheckURLTo = "euc-kr",
```

```
        CheckURLFrom = "utf-8",
        SSIMaxDepth = 16
```

6.1.1.3. history (hist)

사용된 명령어를 보여준다.

- 사용법

```
> history
```

- 예제

```
$$5 webtob (wsadm) [2016-02-03T15:14:47]: history
        5: history
        4: ci -s
        3: ci
        2: ci -s
        1: ci
```

6.1.1.4. !

직전 명령어를 반복한다. !n과 같이사용하면 history에 나타난 과거 명령어 중 특정 명령어를 (n) 반복 수행한다.

- 사용법

```
> !
```

- 예제

```
$$6 webtob (wsadm) [2016-02-03T15:14:49]: !4 ci -s
       Clients  Unique IPs  Dropped
       -------  ----------  -------
HTH 0        0           0        0
  All        0           0        0
```

6.1.2. 동작 상태 정보

6.1.2.1. cliinfo (ci)

현재 접속된 클라이언트(주로 웹 브라우저)의 환경정보를 조회한다. 현재 상태(status), 접속 IP 주소, 처리 건수(count)와 같은 정보를 확인할 수 있다.

- 사용법

```
> ci [-s][-S][-vh Virtual Host 이름][-h HTH 번호]
```

옵션	설명
[-s]	전체 연결된 클라이언트들의 간단한 통계값을 출력한다.
[-S]	연결된 각각의 클라이언트 정보를 출력한다.
[-vh *Virtual Host* 이름]	지정한 Virtual Host에 연결된 클라이언트들의 정보를 출력한다.
[-h *HTH* 번호]	지정한 HTH에 연결된 클라이언트들의 정보를 출력한다.

- 예제
  - 옵션 없이 사용하는 경우

    ci를 옵션 없이 사용하면 다음 같은 결과를 출력한다. 하나의 항목이 1개의 클라이언트를 의미한다.

```
$$1 webtob (wsadm) [2016-02-03T15:14:56]: ci

HTH   0:   RDY
--------------------------------------------------------------------------------
 no   status count idle    local_ipaddr:port    remote_ipaddr:port  spri   user ssl
--------------------------------------------------------------------------------
    0    RDY     0    2    172.16.1.107:8080     172.16.1.100:1951    -1         N
    0    RUN     4    0    172.16.1.107:8080     172.16.1.202:60572   24         N
    1    QED     1    0    172.16.1.107:8080     172.16.1.202:60600   -1         N
    2    RUN     4    0    172.16.1.107:8080     172.16.1.202:60575   26         N
    3    RUN     0    0    172.16.1.107:8080     172.16.1.202:60601   28         N
    4    RUN     4    0    172.16.1.107:8080     172.16.1.202:60577   22         N
    5    RUN     0    0    172.16.1.107:8080     172.16.1.202:60594   23         N
    6    QED     1    0    172.16.1.107:8080     172.16.1.202:60598   -1         N
    7    RUN     0    0    172.16.1.107:8080     172.16.1.202:60596   25         N
    8    QED     1    0    172.16.1.107:8080     172.16.1.202:60597   -1         N
    9    QED     2    0    172.16.1.107:8080     172.16.1.202:60595   -1         N
   10    RUN     4    0    172.16.1.107:8080     172.16.1.202:60584   21         N
   11    QED     2    0    172.16.1.107:8080     172.16.1.202:60592   -1         N
   12    RUN     3    0    172.16.1.107:8080     172.16.1.202:60587   29         N
   13    RUN     4    0    172.16.1.107:8080     172.16.1.202:60588   30         N
   14    QED     1    0    172.16.1.107:8080     172.16.1.202:60599   -1         N
   15    RUN     1    0    172.16.1.107:8080     172.16.1.202:60593   27         N
--------------------------------------------------------------------------------
 HTH     RDY   QED   RUN   ETC total
   0       0     6    10     0    16
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
 HTH     RDY   QED   RUN   ETC total
--------------------------------------------------------------------------------
   0       0     6    10     0    16
--------------------------------------------------------------------------------
 Total     0     6    10     0    16
--------------------------------------------------------------------------------
```

    다음은 출력 항목에 대한 설명이다.

출력항목	설명
status	서버 내부의 클라이언트 상태이다. ◦ RDY : 클라이언트로부터 요청을 받는 중이다. ◦ RUN : 클라이언트의 요청이 서버에서 처리 중이다. ◦ QED : 클라이언트로부터 요청을 받고 처리해야 하는 서버는 확인되었지만 해당 서버가 모두 요청을 처리 중인 상태여서 서버 내 큐에 잠시 대기 중인 상태이다.
count	해당 클라이언트가 전송한 요청 수이다.
idle	해당 클라이언트가 어떠한 데이터도 주고받지 않고 있는 상태로 지속된 시간이다.
local_ipaddr:port, remote_ipaddr:port	서버와 클라이언트 IP:PORT이다.
spri	서버와 클라이언트 IP 주소이다.
user	해당 클라이언트가 일반적인 HTTP 클라이언트가 아닌 경우 어떠한 종류로 사용되고 있는지 알려주는 정보이다. ◦ tcpgw-c : TCPGW 처리 중인 클라이언트이다. ◦ tpcgw-s : TCPGW 처리 중인 서버이다. ◦ conn-c : CONNECT 메소드 요청 처리 중인 클라이언트이다. ◦ conn-s : CONNECT 메소드 요청 처리 중인 서버이다. ◦ rproxy-s : Reverse Proxy에 연결된 서버이다. ◦ rproxy-ws-c : Reverse Proxy에서 WebSocket 사용 중인 클라이언트이다. ◦ rproxy-ws-s : Reverse Proxy에서 WebSocket 사용 중인 서버이다. ◦ internal-c : 내부 redirect 등에 사용 중인 클라이언트이다.
ssl	해당 클라이언트가 SSL로 연결되어 있는지 여부이다.

◦ -s 옵션 사용한 경우

다음과 같이 -s를 사용하면 전체 클라이언트 수와 고유 IP(unique IP) 수, HTH에 등록되지 못하고 연결 종료된 클라이언트 수가 출력된다.

```
$$1 webtob (wsadm) [2016-02-03T15:15:23]: ci -s
       Clients  Unique IPs  Dropped
       -------  ----------  -------
HTH 0       16           0        0
  All       16           0        0
```

6.1.2.2. svrinfo (si)

현재 동작 중인 각 서버의 정보를 조회한다.

- 사용법

```
> si [서버 이름, 서버 이름,..]
```

옵션	설명
[서버 이름, 서버 이름,..]	모든 서버나 지정된 서버의 정보를 출력한다.

- 예제

  다음은 명령어를 실행하는 경우 출력되는 정보이다.

```
$$1 webtob (wsadm) [2016-02-03T15:17:20]: si

------------------------------------------------------------------------------------------
 hth   svrname (svri)   status      reqs     count cqcnt    aqcnt qpcnt emcnt rscnt rbcnt
------------------------------------------------------------------------------------------
   0  MyGroup    (  1)   RDY          12        12     0        0     0     0     0     0
   0  cgi        (  2)   RDY           0         0     0        0     0     0     0     0
   0  php        (  3)   RDY         352       352     0        0     0     0     0     0
```

  다음은 출력 항목에 대한 설명이다.

출력항목	설명
svrname	환경설정의 서버 이름이다.
(svri)	내부적으로 관리되는 서버별 인덱스 번호이다.
status	서버 내부의 클라이언트 상태이다. • RDY : 서버가 요청을 처리할 수 있다. WebtoB와 연결된 서버 프로세스들이 존재한다. • NRDY : 요청을 처리할 수 없다. WebtoB와 연결된 서버 프로세스가 없다. • BLK : 서버가 관리자 명령에 따라 suspend된 상태이다. 서버는 요청을 처리할 수 없다.
reqs	해당 서버에 보내진 요청 수이다.
count	요청처리 수이다.
cqcnt	현재 큐에서 대기 중인 요청 수이다.
aqcnt	현재까지 큐에 대기했던 요청 수(cqcnt의 cumulative 값이다)이다.
qpcnt	큐에 대기 중이던 요청이 timeout 또는 qp 명령 등으로 인해 큐에서 제거된 요청 수이다.
emcnt	큐에 대기 중인 요청 수가 MaxQCount를 넘어간 횟수이다.
rscnt	해당 서버의 비정상 종료로 인한 restart 횟수이다.
rbcnt	해당 서버의 rbs 명령을 통한 reboot 횟수이다.

6.1.2.3. stat (st)

실질적인 시스템 동작 상태를 나타내며, 동작 중인 서버 프로세스와 서비스에 대한 정보를 알 수 있다.

서버 프로세스의 현재 상태, 처리 중인 서비스 이름, 처리한 서비스 개수, 서비스에 대한 상태, 서비스 큐에 존재하는 서비스 요청 개수등과 같은 동적인 정보를 확인할 수 있다.

- 사용법

```
> st [-v 서버 이름,서버 이름,..][-j JSV 서버 이름,JSV 서버 이름,..]
     [-p 서버 프로세스 이름,서버 프로세스 이름,..]
     [-rpg Reverse Proxy Group 이름,Reverse Proxy Group 이름,..]
     [-rproxy Reverse Proxy 이름,Reverse Proxy 이름,..]
     [-tcpgw TCPGW 이름,TCPGW 이름,..]
     [-s 서비스 이름,서비스 이름,..][-h HTH 번호]
     [-T][-t]
```

옵션	설명
[-v 서버 이름, 서버 이름,..]	서버들의 상태를 출력한다. svrinfo와 동일하다.
[-j *JSV*서버 이름, *JSV*서버 이름,..]	JSV 서버들의 통계정보를 출력한다.
[-p 서버 프로세스 이름, 서버 프로세스 이름,..]	개별 서버 프로세스들의 상태를 출력한다.
[-rpg *Reverse Proxy Group* 이름, *Reverse Proxy Group* 이름,..]	Reverse Proxy Group의 통계정보를 출력한다.
[-rproxy *Reverse Proxy* 이름, *Reverse Proxy* 이름,..]	Reverse Proxy의 각 커넥션별 상태를 출력한다.
[-tcpgw *TCPGW* 이름, *TCPGW* 이름,..]	TCPGW의 각 커넥션별 상태를 출력한다.
[-s 서비스 이름, 서비스 이름,..]	서비스 상태를 출력한다.
[-h *HTH* 번호]	HTH 프로세스들의 내부 통계자료를 출력한다.
[-T]	HTH 프로세스 내 각 스레드 타입별 내부 통계자료를 출력한다.
[-t]	HTH 프로세스 내 각 스레드들의 내부 통계자료를 출력한다.

- 예제
  - -p 옵션 사용

    다음은 -p 옵션을 사용해서 서버 프로세스들의 정보를 출력한 예제이다.

```
$$1 webtob (wsadm) [2016-02-03T15:43:22]: st -p

HTH 0(23786): RDY
 ------------------------------------------------------------------------------------
-
```

```
 svr_name   svgname     spr_no(pid)  status      reqs     count    avg(rt)  clid svc v
contime
 ------------------------------------------------------------------------------------
-
 php        phpg        220(  23789)   RDY         0         0   0.0000( 0)   -1   - 0
14187
 ...
 MyGroup1   jsvg        120(      0)   RDY         0         0   0.0000( 0)   -1   - 1
14160
              0 jengineid(ZG9tYWluMS9hZG1pblNlcnZlcg==)(domain1/adminServer)
 MyGroup1   jsvg        121(      1)   RDY         0         0   0.0000( 0)   -1   - 1
14153
              0 jengineid(ZG9tYWluMS9hZG1pblNlcnZlcg==)(domain1/adminServer)
 ...
```

다음은 출력 항목에 대한 설명이다.

출력항목	설명
svr_name	환경설정 SERVER 절의 서버 이름이다.
svgname	SVRGROUP 절의 서버 그룹 이름이다.
spr_no	WebtoB 내부에서 할당한 번호이다.
pid	프로세스의 ID이다. (JSV 서버인 경우 각 커넥션별 worker thread가 보내준 ID)
status	현재상태를 나타낸다. ◦ RDY : 프로세스가 새로운 요청을 기다리고 있다. ◦ NRDY : 요청을 처리할 수 없다. WebtoB와 연결된 서버 프로세스가 없다. ◦ RUN : 프로세스가 요청을 처리하고 있다. ◦ BRUN : 프로세스가 요청에 대한 응답을 전송하는 중이다. 하지만 Flow Control로 인해 잠시 대기 중인 상태이다.
reqs	해당 프로세스로 보내진 요청 수이다.
count	해당 프로세스가 처리한 요청 수이다.
avg	평균 처리시간 (초)이다.
(rt)	현재 처리 중인 요청이 사용한 시간이다.
clid	해당 프로세스가 처리 중인 client ID이다.
svc	요청이 속하는 EXT나 URI 절의 서비스 이름이다.
v	JEUS 연결인 경우 WJP(WebtoB-JEUS Protocol) version 정보를 표시한다. 내부 서버 프로세스인 경우 0으로 표시한다.
contime	해당 프로세스가 HTH에 연결을 맺은 후 지난 시간(HTH에 접속된 시간)이다.

◦ 상태 정보를 여러 번 주기적으로 출력

반복적인 명령어 수행은 상태 정보를 모니터링할 뿐 아니라 업무수행에 대한 디버깅에도 많은 도움을 줄 수 있다.

다음은 5초간의 간격을 두고 'st –s'를 30번 수행하라는 의미이다.

```
$$10 webtob (wsadm(wsmon)) [2009/10/22:12:37:56]: r -i 5 -k 30 st -s
```

다음은 5초간의 간격을 두고 'st –p'를 30초간 수행하는 의미이다.

```
$$10 webtob (wsadm(wsmon)) [2009/10/22:12:37:56]: r -i 5 -f 30 st -p
```

◦ -tcpgw 옵션 사용

다음은 -tcpgw 옵션을 사용해서 각 TCPGW 정보를 출력한 예제이다.

```
$$4 webtob (wsadm) [2018-07-19T12:08:08]: st -tcpgw

   ------------------------------------------------------------------------
    hth  (tcpgwi)tcpgwname    count   avg   cons   remote_ipaddr:port
   ------------------------------------------------------------------------
     0    ( 0/ 0)tcpgw1         2    15.7293  0   127.0.0.1:8088
     0    ( 1/ 0)tcpgw2         0    0.0000   0   192.168.0.1:18088
     0    ( 1/ 1)tcpgw2         0    0.0000   0   192.168.0.1:28088
     0    ( 1/ 2)tcpgw2         0    0.0000   0   192.168.0.1:38088
     0    ( 1/ 3)tcpgw2         0    0.0000   0   192.168.0.1:48088
     0    ( 1/ 4)tcpgw2         0    0.0000   0   192.168.0.1:58088
  ...
```

다음은 출력 항목에 대한 설명이다.

출력항목	설명
hth	hth의 번호이다.
tpcgwname	환경설정의 TCPGW 이름이다.
count	TCP connection을 처리한 횟수이다.
avg	TCP connection이 지속된 시간이다.
cons	연결된 connection 수이다.
remote_ipaddr:port	클라이언트의 요청을 처리할 서버의 IP주소와 포트 번호이다.

◦ -T 옵션 사용

다음은 -T 옵션을 사용해서 각 스레스드들의 정보를 출력한 예제이다.

```
$$1 webtob (wsadm) [2016-02-03T15:45:17]: st -T

HTH 0: RDY
   ------------------------------------------------------------------------
   no  thread_type   status   threads        atasks        ptasks    qtasks
   ------------------------------------------------------------------------
    0   ACCESSLOG      RDY          1            18            18         0
    1   WORKER         RDY         12           401           401         0
    2   SENDFILE       RDY          4             6             6         0
  ...
```

다음은 출력 항목에 대한 설명이다.

출력항목	설명
thread_type	각 스레드의 타입이다.
status	현재상태를 나타낸다. ◦ RDY : 스레드가 새로운 작업을 기다리고 있다. ◦ NRDY : 작업을 처리할 수 없다.
threads	thread_type에 해당하는 스레드 수 이다.
atasks	전체 작업 수 이다.
ptasks	현재까지 처리된 작업 수 이다.
qtasks	아직 처리되지 않고 남아있는(큐잉된) 작업 수 이다.

◦ -t 옵션 사용

다음은 -t 옵션을 사용해서 각 스레드들의 정보를 출력한 예제이다.

```
$$1 webtob (wsadm) [2016-02-03T15:44:57]: st -t

HTH 0: RDY
-------------------------------------------------------------------------------
no  thread_id     status   elapsed     atasks     ptasks    qtasks    task_type
-------------------------------------------------------------------------------
 0   ACCESSLOG      RDY          4         18         18         0         NONE
 1   WORKER001      RUN          0         34         33         1     SSLWRITE
 2   WORKER002      RDY         31         34         34         0         NONE
 3   WORKER003      RDY         31         34         34         0         NONE
 4   WORKER004      RDY         31         34         34         0         NONE
 5   WORKER005      RDY         31         34         34         0         NONE
 6   WORKER006      RDY         31         33         33         0         NONE
 7   WORKER007      RDY         31         33         33         0         NONE
 8   WORKER008      RDY         31         33         33         0         NONE
 9   WORKER009      RDY         31         33         33         0         NONE
10   WORKER010      RDY         31         33         33         0         NONE
11   WORKER011      RDY         31         33         33         0         NONE
12   WORKER012      RDY         31         33         33         0         NONE
13   SENDFILE001    RDY          4          2          2         0         NONE
14   SENDFILE002    RDY          4          1          1         0         NONE
15   SENDFILE003    RDY          4          1          1         0         NONE
16   SENDFILE004    RDY          4          2          2         0         NONE
 ...
```

다음은 출력 항목에 대한 설명이다.

출력항목	설명
thread_id	각 스레드의 이름이다.

출력항목	설명
status	현재상태를 나타낸다. ◦ RDY : 스레드가 새로운 작업을 기다리고 있다. ◦ NRDY : 작업을 처리할 수 없다. ◦ RUN : 스레드가 작업을 처리하고 있다.
elapsed	작업시작 또는 작업완료 후 지나간 시간이다.
atasks	전체 작업 수 이다.
ptasks	현재까지 처리된 작업 수 이다.
qtasks	아직 처리되지 않고 남아있는(큐잉된) 작업 수 이다.
task_type	현재 스레드에서 처리 중인 작업이다. 작업을 처리하고 있지 않은 경우 NONE이 출력된다. 스레드가 깨어났으나 아직 작업을 시작하지 않은 경우 status는 RUN이고 task_type은 NONE으로 출력된다.

6.1.3. 서버 프로세스 중지 및 재개

6.1.3.1. suspend (sp)

동작 중인 서버 프로세스의 활동을 중지시킨다. 중지된 서버 프로세스는 현재 처리 중인 서비스를 정상 완료한 후 더 이상의 동작은 중지하고, 큐에 있는 서비스들은 대기상태가 된다.

응용 서버 프로그램 오류 등으로 더 이상의 업무 처리가 불가능하여 이를 해결하기 위해 동작 중인 서버 프로세스를 중지시킬 필요가 발생할 수 있다. 이와 같이 원인을 알 수 없는 문제로 인하여 요청 처리를 중지시키고 추가적인 작업을 필요로 하는 경우에 유용하게 사용할 수 있는 기능이다.

- 사용법

```
> suspend [-v 서버 이름]
```

옵션	설명
[-v 서버 이름]	지정된 서버를 중지한다.

- 예제

  다음은 PHP 서버를 중지하는 예제이다. "st -v"를 사용해서 서버 상태를 확인하면 BLK으로 변경되었음을 확인한다.

```
$$1 webtob (wsadm) [2016-02-03T15:56:57]: suspend -v php
Server(php) is supended
$$2 webtob (wsadm) [2016-02-03T15:57:07]: st -v

------------------------------------------------------------------------------------------
 hth   svrname (svri)   status      reqs     count cqcnt    aqcnt qpcnt emcnt rscnt rbcnt
------------------------------------------------------------------------------------------
```

```
   ...
   0  php        (  3)   BLK         539       539     0      403     0     0     0     0
   ...
```

6.1.3.2. resume (rs)

suspend로 의해 동작 중지된 서버 프로세스의 활동을 재개시킨다. 활동이 재개된 서버 프로세스는 큐에 대기 중이던 서비스를 처리하기 시작하며 요청되는 서비스에 대해 처리 가능 상태가 된다.

- 사용법

```
> resume [-v 서버 이름]
```

옵션	설명
[-v 서버 이름]	지정된 서버를 처리 가능 상태로 변경한다.

- 예제

  다음은 중지된 PHP 서버를 재개하는 예제이다. "st -v"를 사용해서 서버 상태를 확인하면 RDY로 변경된다.

```
$$3 webtob (wsadm) [2016-02-03T15:57:27]: st -v resume -v php
Server(php) is resumed
$$4 webtob (wsadm) [2016-02-03T15:57:37]: st -v
------------------------------------------------------------------------------------------
 hth   svrname (svri)   status      reqs     count cqcnt    aqcnt qpcnt emcnt rscnt rbcnt
------------------------------------------------------------------------------------------
   ...
   0  php        (  3)   RDY         539       539     0      403     0     0     0     0
   ...
```

6.1.4. 적체 해소

6.1.4.1. qpurge (qp)

업무의 폭주 현상이 발생하여 많은 업무가 적체되어 정상적으로 거래를 처리하지 못하는 경우 현재 큐에 적체되어 있는 서비스 요청을 삭제해서 원활한 업무수행을 유도하는 기능이다. 하루에도 수십만 건의 업무를 처리하는 은행이나 관공서에서 유용하게 사용될 수 있는 기능으로서 삭제된 업무는 클라이언트의 재요청을 통하여 다시 처리될 수 있다. WebtoB에서는 서버 프로세스별로 큐를 관리함으로써 관리자는 특정 서버별로 큐를 삭제할 수 있다. 따라서 특정 업무별로 처리가 가능하며 이는 타 업무의 효과적인 수행에도 도움을 줄 수 있다.

qp를 통해 삭제된 서비스는 클라이언트에게 다음의 에러를 전송한다.

```
503 Service Temporarily Unavailable.
```

- 사용법

```
> qpurge [-v 서버 이름]
```

옵션	설명
[-v 서버 이름]	지정된 서버의 큐에 남아있는 요청들을 제거한다.

- 예제

  다음은 PHP 서버의 큐에 대기하고 있는 요청들을 제거하는 예제이다. purged_count는 6개의 요청이 제거되었다는 결과를 출력한다.

```
$$11 webtob (wsadm) [2016-02-03T15:58:52]: qpurge -v php
q for svr php is purged: purged_count = 6
```

6.1.5. 설정값 동적 변경

6.1.5.1. set

현재 설정되어 있는 환경 파일의 설정값을 동적으로 변경할 수 있는 명령어로 사용법은 다음과 같다. 변경 가능 항목은 wsadmin의 cfg 명령어를 통해서 확인할 수 있다.

- 사용법

```
> set [-n 노드 이름][-vh VHOST 이름][-g SVRGROUP 이름][-v 서버 이름][-s 서비스 이름]
```

옵션	설명
[-n 노드 이름]	NODE 절을 설정한다.
[-vh *VHOST* 이름]	지정한 이름의 VHOST를 설정한다.
[-g *SVRGROUP* 이름]	지정한 이름의 서버 그룹을 설정한다.
[-v 서버 이름]	지정한 이름의 서버를 설정한다.
[-s 서비스 이름]	지정한 이름의 서비스를 설정한다.

- 예제

  다음은 "cfg -v php"를 사용해서 MaxQCount 값을 확인한 후 set을 사용해서 값을 변경하는 예제이다. set 명령 수행 후 값이 100으로 변경된 것을 확인할 수 있다.

```
$$10 webtob (wsadm) [2016-02-03T16:01:32]: cfg -v php
    SERVER(3): Name = php,
         SvgName = phpg,
         MinProc = 10,
         MaxProc = 10,
         MaxQCount(mq) = 0,
         ...
$$11 webtob (wsadm) [2016-02-03T16:01:42]: set -v php mq 100
new value (100) is set for section = SERVER, name = php, fld = mq
```

```
$$12 webtob (wsadm) [2016-02-03T16:01:52]: cfg -v php
    SERVER(3): Name = php,
         SvgName = phpg,
         MinProc = 10,
         MaxProc = 10,
         MaxQCount(mq) = 100,
         ...
```

6.1.6. 클라이언트 연결해제

6.1.6.1. discon (ds)

현재 접속되어 있거나 아무 일도 수행하지 않는 클라이언트를 강제로 연결을 해제할 수 있다. 클라이언트 정보를 얻을 수 있는 ci 명령어로 확인 후 사용한다.

- 사용법

```
> discon [-h HTH 번호] -a | -i idle time | -c Client ID [-f]
```

옵션	설명
[-h *HTH* 번호]	지정된 HTH에 연결된 클라이언트 연결을 끊는다.
-a	모든 HTH에 연결된 클라이언트 연결을 끊는다.
-i *idle time*	지정된 시간(초) 이상 경과된 클라이언트 연결을 끊는다.
-c *Client ID*	지정된 클라이언트(ci가 출력한 클라이언트 ID)를 끊는다.
[-f]	즉시 연결을 해제한다.

- 예제

  다음은 ci 명령어로 클라이언트 ID를 확인한 후 discon 명령어를 사용해서 15번 클라이언트의 연결을 강제로 끊는 예제이다.

```
$$16 webtob (wsadm) [2016-02-03T16:02:42]: ci
HTH   0:   RDY
--------------------------------------------------------------------------------
 no   status count idle    local_ipaddr:port    remote_ipaddr:port  spri   user
--------------------------------------------------------------------------------
   15    RDY     0    2    172.16.1.107:8080     172.16.1.202:36505   -1
   ...

$$17 webtob (wsadm) [2016-02-03T16:02:55]:  discon -c 15
client (hth0: 15) is disconnected
```

6.1.7. 기타

6.1.7.1. cachelist

현재 WebtoB의 HTTP 응답 캐시에 저장된 응답들에 대한 정보를 출력한다. 정보량이 크기 때문에 결과물은 별도 파일에 저장된다.

- 사용법

```
> cachelist
```

- 예제

  다음은 요청 "/index.html"에 대한 응답이 캐시에 저장된 것을 보여주는 예제이다. 요청 경로 외 부분은 서버 내부 디버그 용도로만 사용된다.

```
$$1 webtob (wsadm) [2016-02-03T16:02:15]:  cachelist
Cache contents files are created in /root/wb-5000-clean/log/cachelist/. Please check the
directory.

생성된 파일 예제:
0 webtob:8080/index.html 5 0 2016/02/03:14:59:45 10 0 327043 5 2016/02/03:15:48:56 (1253083736)
Total cached response=1
Total content length=5
```

6.1.7.2. cacherefresh (cr)

WebtoB의 HTTP 응답 캐시에 저장된 응답들을 제거한다.

- 사용법

```
> cacherefresh {-a | -h | -i | -j | -r | -u URL}
```

옵션	설명
[-a]	캐시된 모든 응답을 제거한다.
[-h]	SVRTYPE이 HTML로 처리된 후 캐시된 HTML과 유사 텍스트 응답만 제거한다.
[-i]	SVRTYPE이 HTML로 처리된 후 캐시된 IMAGE 타입 응답만 제거한다.
[-j]	SVRTYPE이 JSV으로 처리된 후 캐시된 응답만 제거한다.
[-r]	Reverse Proxy로 처리된 후 캐시된 응답만 제거한다.
[-u *URL*]	캐시된 응답 중 URL을 지정하여(fnmatch 방식으로) 매칭된 응답만 제거한다.

- 예제

  다음은 캐시에 저장된 모든 응답을 제거하는 예제이다.

```
$$1 webtob (wsadm) [2016-02-03T16:04:57]:  cacherefresh -a
```

다음은 캐시에 저장된 응답 중 "test.domain.com/test.html"을 제거하는 예제이다.

```
$$2 webtob (wsadm) [2016-02-03T16:05:07]:  cacherefresh -u test.domain.com/test.html
```

6.1.7.3. clilisten (cl)

WebtoB가 클라이언트 연결을 받기 위해서 사용하는 Listen Port들을 제거, 생성할 수 있다.

- 사용법

```
> clilisten {on | off}
```

옵션	설명
on	Listen Port들을 생성한다.
off	Listen Port들을 제거한다. 제거되면 새로운 클라이언트가 WebtoB로 접속할 수 없다. 이미 HTH와 연결된 클라이언트에 대해서는 (처리 중인 요청을 완료한 후) 연결을 종료한다.

- 예제

  다음은 Listen Port들을 제거하는 예제이다.

```
$$18 webtob (wsadm) [2016-02-03T16:05:17]: clilisten off
client listen blocked
```

  다음은 Linux 콘솔에서 Listen Port들을 확인한 결과이다. Listen Port는 8080으로 가정한다.

```
[root@webtob ~]# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:9090                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:681                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:8080                0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
...
```

  clilisten off를 실행한 후 다시 포트를 확인한다. 8080 포트가 제거된 것을 확인한다.

```
[root@webtob ~]# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:9090                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:681                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
...
```

6.1.7.4. logstart (logs), logend (loge)

실행한 명령들과 출력물을 로그 파일에 기록한다. 로그 파일은 현재 디렉터리에 생성된다.

로그 데이터를 통해 업무 폭주 시간, 불필요한 서버 프로세스, 큐잉 상태 등을 확인하여 전반적인 시스템 분석이 가능하다. logstart가 logging을 시작하고, logend가 logging을 중단한다.

- 사용법

```
> logstart (logs)
```

```
> logend (loge)
```

- 예제

  다음 "log.txt" 파일에 st -p의 결과값이 저장된다.

```
$$1 webtob (wsadm) [2016-02-03T16:05:55]: logstart log.txt
logging start ok
$$2 webtob (wsadm) [2016-02-03T16:05:59]: st -p
...
$$3 webtob (wsadm) [2016-02-03T16:06:05]: logend
logging end ok
$$4 webtob (wsadm) [2016-02-03T16:06:25]:
```

6.1.7.5. logsearch

지정한 로그 파일의 내용을 검색할 수 있다.

- 사용법

```
> logsearch [keyword] [logfile]
```

옵션	설명
[*keyword*]	로그 파일 내에서 검색할 문자열이다. 검색시 대소문자는 무시된다.
[*logfile*]	문자열을 검색할 로그 파일이다. 절대 경로 및 상대 경로를 지원한다. • 상대 경로로 입력한 경우 syslog가 저장되는 폴더가 기준 경로이다. • 절대 경로로 입력하는 경우 권한이 있는 모든 파일에 접근이 가능하기 때문에 사용에 유의해야 한다.

- 예제

  다음은 syslog의 내용을 검색하는 예제이다.

```
$$1 webtob (wsadm) [2020-08-05T14:58:35]: logsearch successfully system.log_08052020
[2020-08-05T10:44:16] [HTH(22378)] [I] [HTH-00074] Successfully connected and registered to HTL.
```

```
HTH is ready to receive client connections.
[2020-08-05T10:44:16] [HTL(22377)] [I] [HTL-00021] Successfully registered HTH. Start listening
for this HTH. index=0, fd=6
[2020-08-05T14:57:29] [HTH(4901)] [I] [HTH-00074] Successfully connected and registered to HTL.
HTH is ready to receive client connections.
[2020-08-05T14:57:29] [HTL(4900)] [I] [HTL-00021] Successfully registered HTH. Start listening
for this HTH. index=0, fd=6
```

6.1.7.6. ll

LOGLEVEL 설정을 동적으로 변경할 수 있다. 로거 이름, 레벨, 옵션은 설정 항목을 참고한다.

- 사용법

```
> ll [로거 이름][-l level | -o option | -rs seconds | -rf fileSize]
```

옵션	설명
[로거 이름]	WebtoB가 지원하는 로거를 설정한다.
[-l *level*]	로그 레벨을 지정된 level로 변경한다.
[-o *option*]	로거의 옵션을 option으로 변경한다.
[-rs *seconds*]	로거의 rotateBySeconds 설정을 seconds로 변경한다.
[-rf *fileSize*]	로거의 rotateByFileSize 설정을 fileSize로 변경한다.

- 예제

  다음 명령은 ".hth" 로거의 레벨을 DEBUG로 변경한다. 변경된 값은 cfg -ll로 확인할 수 있다.

```
$$6 webtob (wsadm) [2016-02-03T16:06:35]: ll .hth -l DEBUG
Log level is successfully updated. logger=.hth, level=DEBUG, options=
```

6.1.7.7. patchinfo

릴리즈된 이후 추가 패치를 적용한 경우 패치된 정보를 보여준다.

- 사용법

```
> patchinfo
```

- 예제

  다음은 릴리즈된 버전에서 patchinfo 명령을 실행한 예제이다.

```
$$1 webtob (wsadm) [2016-02-03T16:07:35]: patchinfo

----------
Released (2015/01/29) : 5.0.0.0-B41.0.0
```

```
----------
```

6.1.7.8. rebootsvr (rbs)

현재 사용 중인 서버 프로세스를 새로운 프로세스로 변경하는 경우에 사용한다.

WEBTOB_BKAPPDIR을 환경변수에 지정하고 새로운 프로그램을 지정된 디렉터리에 옮기고 다음의 명령어를 수행한다.

- 사용법

```
> rebootsvr new_file old_file
```

옵션	설명
new_file	변경된 파일(object file)을 설정한다.
old_file	현재 실행 중인 파일(object file)을 설정한다.

6.1.7.9. repeat (r)

현재 정보를 일정한 시간 간격을 두고 계속 모니터링하는 경우에 사용한다.

- 사용법

```
> repeat [-i 초] [-k 횟수| -f 초] 명령어
```

옵션	설명
[-i 초]	명령어 반복 주기(초)를 설정한다.
[-k 횟수]	지정한 횟수만큼 반복하고 중지한다.
[-f 초]	지정한 시간(초)이 경과되면 반복을 중지한다.
명령어	반복 수행할 명령어를 설정한다.

- 예제

  다음은 5초간의 간격을 두고 st –s를 30번 수행하도록 하는 예제이다.

```
$$9 webtob (wsadm) [2016-02-03T16:08:02]: r -i 5 -k 30 st -p
```

6.1.7.10. restat

서버 프로세스의 통계 정보를 초기화하는 경우에 사용한다. 해당 명령을 사용하면 관련된 서비스나 서버 프로세스 의 정보가 초기화된다.

- 사용법

```
> restat [-a | -v 서버 이름]
```

옵션	설명
[-a]	모든 서버 프로세스의 통계 정보를 초기화한다.
[-v 서버 이름]	사용자가 지정한 서버 프로세스의 통계 정보를 초기화한다.

- 예제

  다음은 "MyGroup"이라는 이름을 가진 서버 프로세스의 통계 정보를 초기화하는 예제이다.

```
$$1 webtob (wsadm) [2020-08-11T17:25:07]: restat -v MyGroup
```

6.2. wsmon 콘솔 모니터링 프로그램

wsmon은 텍스트 기반의 모니터링 환경을 제공한다. 항상 프롬프트(prompt) 상태로 대기 중이다가 입력되는 명령어를 해석하여 이를 실행하게 된다.

wsmon 프로그램은 다음과 같이 실행한다.

```
$ wsmon
```

이후 아래와 같은 메시지와 함께 프롬프트(prompt)가 나타난다. 이것이 wsmon 프로그램 실행 상태임을 나타낸다.

```
--- Welcome to WebtoB Mon (Type "quit" to leave) ---
$$1 webtob (wsmon) [2016-02-03T16:08:22]:
```

wsmon 툴에서 사용할 수 있는 명령어들의 도움말을 조회할 때는 **'help <명령어>'**가 사용된다.

```
$$5 webtob (wsmon) [2016-02-03T16:08:29]: help ci
Summary: cliinfo, ci: show client properties
Usage: cliinfo (ci) [-s | -S] [-vh vhostname] [-h hthno]
            -s : summury of the client info
            -S : detail summury of the client info
            -vh vhostname : specify VHOST
            -h hthno : specify a HTH number to view
```

wsmon 툴을 종료하기 위해서는 quit (q) 명령을 사용한다.

```
$$1 webtob (wsmon) [2016-02-03T16:08:42]: quit
```

다음은 wsmon이 제공하는 명령어들이다. 자세한 설명은 각 해당 절의 설명을 참고한다.

명령어	약자	설명
cachelist		HTTP 응답 캐시에 저장된 응답들의 정보 출력한다.

명령어	약자	설명
cliinfo	(ci)	접속 웹 브라우저를 확인한다.
config	(cfg)	환경설정 내용을 조회한다.
history	(hist)	최근 수행한 50개 명령을 조회한다.
help	(h)	도움말을 조회한다.
logend	(loge)	logging을 종료한다.
logstart	(logs)	logging을 시작한다.
repeat	(r)	명령어를 반복한다.
stat	(st)	프로세스 및 서비스 상태에 대한 통계를 조회한다.
svrinfo	(si)	서버 정보를 확인한다.
webtobinfo	(wi)	WebtoB 시스템의 정보를 확인한다.
patchinfo		릴리즈된 이후 패치된 정보를 조회한다.
quit	(q)	wsmon을 종료한다.
!		직전 명령어를 반복한다.

wsmon 명령어의 기본 사용법은 **wsadmin 콘솔 관리자 프로그램**과 동일하고 일부 출력되는 정보에만 차이가 있다. 따라서 각 명령어의 사용법에 대한 자세한 설명은 wsadmin 콘솔 관리자 프로그램을 참고한다.

6.3. 명령어

다음은 WebtoB에서 제공되는 명령어이다.

옵션	설명
wscfl	텍스트 형태의 WebtoB 환경 파일을 컴파일하여 wsconfig(이진 WebtoB 환경 파일)을 생성하는 명령어이다.
wsuncfl	컴파일된 wsconfig(이진 WebtoB 환경 파일)을 다시 역으로 분석하여 텍스트 형태의 환경 파일을 생성하는 명령어이다.
wsgst	이진 WebtoB 환경 파일을 참조하여 서비스 테이블을 생성하는 명령어이다.
wsracd	WebAdmin을 지원하기 위한 명령어이다.
wsboot	WebtoB 시스템의 전체나 또는 일부분을 실행시키는 명령어이다.
wsdown	WebtoB 시스템 전체나 또는 일부분을 종료시키는 명령어이다.

6.3.1. wscfl

wscfl은 텍스트 형태의 WebtoB 환경 파일을 컴파일하여 wsconfig(이진 WebtoB 환경 파일)을 생성하는 명령어로 WebtoB 시스템이 설치된 운영 시스템 환경에서 사용할 수 있다.

입력 파일을 컴파일하는 중에 에러가 발견되면, 이진 WebtoB 환경 파일을 생성하지 않고 컴파일을 중단하게 된다.

에러가 발견되지 않으면 이진 파일로 변환된 WebtoB 환경 파일이 생성된다. 이 명령어로 생성된 WebtoB 환경 파일은 wsboot, wsdown 등에서 사용한다.

- 사용법

```
$ wscfl -i 텍스트 WebtoB 환경파일이름 [-o 이진 WebtoB 환경파일이름][-b][-v]
```

옵션	설명
-i 텍스트 *WebtoB* 환경파일이름	컴파일 대상이 되는 환경 파일(텍스트 형태의 WebtoB 환경파일이름)을 명시하는 데 사용되며, 반드시 필요한 옵션이다. 경로와 함께 지정될 수 있으며 지정한 환경 파일을 찾지 못한 경우에는 경고 메시지를 출력한다.
[-o 이진 *WebtoB* 환경파일이름]	컴파일 결과물인 이진 WebtoB 환경파일이름을 명시하는 데 사용된다. 선택 옵션으로 경로와 함께 지정할 수 있으며 경로가 지정되지 않은 경우 현재 wscfl 명령을 실행 중인 디렉터리에 명시된 이름으로 컴파일된 결과물이 생성된다. 옵션이 생략되면 기본값으로 wsconfig라는 이름의 파일로 생성된다.
[-b]	WebtoB의 환경설정 파일을 백업한다.
[-v]	WebtoB의 버전을 확인하는 데 사용된다.

- 예제
  - [-i] 옵션 사용

    다음은 디렉터리의 http.m이란 텍스트 형태의 WebtoB 환경 파일을 컴파일하여 현재 디렉터리에 기본으로 wsconfig라는 이진 WebtoB 환경 파일을 생성하는 예제이다.

    ```
    $ wscfl -i http.m
    ```

    다음은 디렉터리의 http.m라는 텍스트 형태의 WebtoB 환경 파일을 컴파일하여 현재 디렉터리에 wsconfig2라는 이름으로 이진 WebtoB 환경 파일을 생성하는 예제이다.

    ```
    $ wscfl -i http.m -o wsconfig2
    ```

  - [-v] 옵션 사용

    다음은 WebtoB의 버전을 확인하는 예제이다.

    ```
    $ wscfl -v
    ```

6.3.2. wsuncfl

wsuncfl는 컴파일된 wsconfig(이진 WebtoB 환경 파일)을 다시 역으로 분석하여 텍스트 형태의 환경 파일을

만드는 명령어로 WebtoB 시스템이 설치된 운영 시스템 환경에서 사용할 수 있다.

wscfl로 컴파일되어 이진 파일로 변한 환경설정 파일을 알아볼 수 있는 텍스트 형태로 변환한다. 환경 파일의 원본을 잃어버리고 이진 파일만 가지고 있는 경우 wsuncfl을 사용해서 원래의 텍스트 형태의 환경 파일을 복구할 수 있다.

- 사용법

```
$ wsuncfl [-i 이진 WebtoB 환경파일이름] [-o 텍스트 WebtoB 환경파일이름] [-v]
```

옵션	설명
[-i 이진 *WebtoB* 환경파일이름]	디컴파일 대상이 되는 바이너리 형태의 환경 파일의 이름을 명시하는 데 사용된다. 선택 옵션으로 경로와 함께 지정할 수 있으며, 지정한 환경 파일을 찾지 못한 경우에는 경고 메시지를 출력한다. 옵션이 생략되면 기본으로 wsconfig라는 이름이 사용된다.
[-o 텍스트 *WebtoB* 환경파일이름]	디컴파일 결과물인 텍스트 WebtoB 환경파일이름을 명시하는 데 사용된다. 반드시 필요한 옵션으로 경로와 함께 지정할 수 있으며, 경로가 지정되지 않은 경우 현재 wsuncfl 명령을 실행 중인 디렉터리에 명시된 이름으로 디컴파일된 결과물이 생성된다.
[-v]	WebtoB의 버전을 확인하는 데 사용된다.

- 예제

  다음은 디렉터리의 wsconfig라는 이름의 이진 WebtoB 환경 파일을 현재 디렉터리에 sample_r.m이라는 텍스트 형태의 WebtoB 환경 파일로 디컴파일하는 예제이다.

```
$ wsuncfl –o sample_r.m
```

6.3.3. wsgst

wsgst는 이진 WebtoB 환경 파일을 참조하여 서비스 테이블을 생성하는 명령어로 WebtoB 시스템이 설치된 운영 시스템 환경에서 사용할 수 있다.

wsgst는 wscfl 명령어에 의해서 생성된 이진 WebtoB 환경 파일의 SERVER 절과 SERVICE 절을 참조하여 각 서버별로 서비스 테이블을 생성한다. 이 테이블은 서버 프로세스에서 제공하는 서비스 명단으로써 서버 프로그램을 작성하여 컴파일할 때 함께 컴파일되어 서버 프로세스가 동작할 때 서비스 위치를 찾는 데 사용된다.

wsgst 명령의 결과는 WEBTOBDIR로 지정된 디렉터리 하위의 'svct' 디렉터리에 '서버 이름_svctab.c'의 이름으로 생성된다. 서버 이름은 SERVER 절에 등록된 서버 이름이며, 각 파일의 내용은 SERVICE 절에 등록된 해당 서버가 제공하는 서비스 이름들이다.

WebtoB 환경 파일에 등록된 서비스는 반드시 해당 서버의 서비스 테이블에도 등록이 되어야 한다. 즉, WebtoB 환경 파일의 SERVER 절이나 SERVICE 절에 서버 이름, 서비스 이름, 서비스의 SVRNAME 등이 변경된 경우 wsgst 명령을 실행한 후 그 결과물인 서비스 테이블과 함께 서버 프로그램을 다시 컴파일해야 한다.

- 사용법

```
$ wsgst [-f 이진 WebtoB 환경파일이름]
```

옵션	설명
[-f 이진 *WebtoB* 환경파일이름]	참조할 이진 WebtoB 환경파일이름을 명시하는 데 사용된다. wscfl의 결과물로써 wsboot와 wsdown에서도 참조되는 파일이다. 경로와 함께 지정될 수 있으며, 이 옵션이 생략되면 기본으로 wsconfig라는 이름이 사용된다.

- 예제

  다음은 WEBTOBDIR로 지정된 디렉터리 하위의 config 디렉터리에서 wsconfig을 참조하여 서비스 테이블을 생성하는 예제이다.

```
$ wsgst
```

  다음은 /user1/park/WebtoB/bin 디렉터리의 wsconfig2 환경 파일을 참조하여 서버별로 서비스 테이블을 생성하는 예제이다.

```
$ wsgst -f /user1/park/webtob/config/wsconfig2
```

  예를 들어 환경 파일을 다음 페이지와 같이 등록한 경우 /user1/park/webtob/svct 디렉터리에 webaps_svctab.c라는 서비스 테이블이 생성된다.

```
...
*SERVER
webaps          SVGNAME=webapsg

*SERVICE
svc1            SVRNAME=webaps
svc2            SVRNAME=webaps
                        ...
```

6.3.4. wsracd

wsracd는 WebAdmin 및 SysMaster를 지원하기위한 명령어로 WebAdmin 또는 SysMaster를 실행하기 전에 반드시 wsracd를 실행시켜야 한다. wsracd는 WebtoB 시스템이 설치된 운영 시스템 환경에서 사용할 수 있다(WebAdmin은 현재는 향후 지원 예정이다).

- 사용법

```
$ wsracd [-k] [-f 이진 WebtoB 환경파일이름] [-a] [-d]
```

옵션	설명
[-k]	이진 WebtoB 환경 파일을 참조할 것인지 여부를 지정하는 옵션이다. 옵션을 지정하면 이진 WebtoB 환경 파일을 참조하지 않고 기본 포트로 설정되어 있는 포트를 사용한다. 환경변수로 WEBTOB_RAC_PORT를 설정하였다면 환경변수로 설정된 포트를 사용한다. 보통 wsracd는 이 옵션을 이용하여 실행한다.
[-f 이진 *WebtoB* 환경파일이름]	사용자가 지정한 이진 WebtoB 환경 파일을 참조하여 실행한다. 사용자가 지정한 이진 WebtoB 환경 파일에 설정된 포트를 사용한다.
[-a]	anonymous(no authentication) mode로 실행한다. 해당 옵션을 사용하지 않는 경우 OS의 계정을 통해 wsracd에 connect할때 인증절절차를 거쳐야 한다. OS의 계정을 사용하여 인증하기 때문에 ${WEBTOBDIR}/bin/wsracd를 root 계정으로 변경하고, sticky bit을 설정해야 한다.
[-d]	debug mode로 실행한다.

- 예제

  이진 WebtoB 환경 파일을 참조하여 실행한다. 옵션을 사용하지 않으면 기본으로 ${WEBTOBDIR}/config/wsconfig를 참조한다. wsconfig 파일이 존재하지 않으면 에러 메시지를 출력한다.

```
$ wsracd -a
```

  다음은 환경 파일을 참조하지 않고 다른 명령어(wsboot)에서 사용한 정보만 이용하는 예제이다.

```
$ wsracd –k
```

  다음은 사용자가 지정한 /user1/park/webtob/config/wsconfig2라는 이진 WebtoB 환경 파일을 참조하여 실행하는 예제이다.

```
$ wsracd –f /user1/park/webtob/config/wsconfig2
```

6.3.5. wsmkppd

wsmkppd는 SSL(PROXY_SSL)절의 PasePhraseDialog를 지원하기위한 명령어이다.

SSL(PROXY_SSL) 절에 암호화된 개인키를 설정한 경우 WebtoB를 기동할 때 암호문 입력을 요구한다. 기동할 때마다 암호문을 입력하는 것은 번거롭기 때문에 PassPhraseDialog 설정해서 사용하도록 한다. wsmkppd는 PassPhraseDialog에 적용할 수 있도록 passphrase 암호를 저장하고있는 passphrase file을 생성한다. PassPhraseDialog 사용법은 SSL 절과 PROXY_SSL 절을 참고한다.

- 사용법

```
$ wsmkppd [-p passwd] ppd_filename sslname
```

옵션	설명
[-p *passwd*]	암호화된 개인키에 대한 암호문을 입력한다.
ppd_filename	PassPhraseDialge에 사용되는 파일명으로 wsmkppd 실행의 결과를 해당 파일에 (추가)저장하게 된다.
sslname	SSL(PROXY_SSL) 절에 설정한 name이다.

- 예제

  -p 옵션에서 암호문을 사용하지 않으면 기본으로 암호문 입력을 요구한다.

  - 다음은 wsmkppd 실행 시 옵션을 주지 않은 결과이다.

    ```
    $ wsmkppd
    Usage: wsmkppd [-p passwd] ppd_filename sslname
             -p passwd: password of private key.
             ppd_filename: output file for PassPhraseDialog.
             sslname: name of SSL(PROXY_SSL) section.
    ```

  - 다음은 wsmkppd를 실행할 때 ppd_filename과 sslname만 주고 실행한 결과이다.

    ```
    $ wsmkppd ssl.ppd ssl1
    New password: (암호문 입력)
    Confirm: (암호문 재입력)
    Adding Password for ssl1

    $ls -al ssl.ppd
    -rw-------  1 webtob webtob 19 Feb 19 10:33 ssl.ppd
    ```

  - 다음은 wsmkppd를 실행할 때 -p 옵션으로 암호문(mypasswd)을 주고 실행한 결과이다(ppd_filename이 이미 존재하면 내용만 추가하게 된다).

    ```
    $ wsmkppd -p mypasswd ssl.ppd ssl2
    Adding Password for ssl2

    $ls -al ssl.ppd
    -rw-------  1 webtob webtob 42 Feb 19 10:42 ssl.ppd
    ```

  - 다음은 wsmkppd 결과로 생성된 파일의 내용 예제이다.

    ```
    $ cat ssl.ppd
    ssl1:gJDP+OmN+wI=:
    ssl2:8eG+iZjNiu5k:
    ```

    다음은 SSL 절 설정 예제이다.

```
*SSL
ssl1          CertificateFile = "/webtob/ssl/newcert.pem",
              CertificateKeyFile = "/webtob/ssl/newcert.pem",
              PassPhraseDialog = "file:/webtob/ssl/ssl.ppd"
```

7. WebtoB 튜닝

WebtoB는 상업용 웹 서버로 대규모의 전문적인 웹 서비스의 운영을 위해 개발되었다. 전문가를 위한 서버인 만큼 사용자에 대한 자유도가 높고 각자의 환경에 맞는 최적의 미세조정을 제공한다. 웹 서버 운영자가 각자의 환경에 맞게 서버를 튜닝하여 사용하게 되면 엄청난 효율과 성능을 얻을 수 있다.

WebtoB의 이러한 특징을 적절히 활용하여 사용할 수 있도록 본 장에서는 튜닝에 있어 가장 중요한 항목들과 그 적용사례에 대해서 설명한다.

7.1. HTH 및 HTH_THREAD 설정

WebtoB는 특정 프로세스에서 일반 브라우저에 관련된 연결을 모두 관리한다. 이러한 구조는 하나의 프로세스에서 모든 처리를 관리하므로 빠른 작업을 할 수 있는 장점이 있으나 프로세스의 안정성에 의해 시스템의 성능이 크게 좌우되므로 주의해야 할 필요가 있다. HTH라는 프로세스가 바로 이런 역할을 한다. 이 HTH라는 프로세스는 멀티 스레드로 구성되어 있기 때문에 이 프로세스 및 스레드 관리가 성능에 가장 큰 영향을 줄 수 있으므로 이를 적당히 조절할 필요가 있다.

7.1.1. HTH 설정

WebtoB에서는 HTH 프로세스를 통해서 사용자의 연결을 관리하고 HTH 프로세스에 모든 사용자의 연결이 맺어진다. HTH 프로세스 하나에 몇 개의 연결을 맺는 것이 좋은가는 사용하는 운영체제마다 다르다. 이는 각 운영체제마다 프로세스 하나가 열 수 있는 연결의 수를 제한하고 있기 때문이다.

또한, 하나의 프로세스에서 너무 많은 처리를 하면 많은 CPU 점유를 차지하게 되고 프로세스의 동작에 무리를 줄 수 있다. 따라서 동시 사용자가 많은 경우 가급적 적당한 정도로 분리하여 실제 사이트에서 운영한다. HTH process 하나당 최대로 처리할 수 있는 연결의 개수는 FD 설정과 관련이 있다. 따라서 더 많은 동시 사용자를 처리하고자 하는 경우 프로세스 하나가 열 수 있는 연결 제한(최대 16384)을 더 크게 하거나, NODE 절의 HTH 항목의 설정(최대 100)을 늘려서 사용할 수 있다.

이러한 정보는 WebtoB 환경 파일을 컴파일하면, 화면에 WebtoB가 자체적으로 운영체제에 문의하여 자신이 받아들일 수 있는 최대의 동시 사용자수를 출력해 준다. 따라서 운영자는 동시 사용자 수의 예측과 이 정보를 바탕으로 HTH 프로세스의 적당한 수를 예측해야 한다.

WebtoB는 HTH 프로세스가 요청 처리의 많은 부분을 관리하기 때문에 운영장비에 WebtoB만 단독으로 사용할 경우 운영장비의 CPU Core 수 만큼 HTH를 설정하는 것이 장비를 효율적으로 사용할 수 있는 방법이며 이하 Thread 설정을 추가적으로 고려할 수 있다. HTH를 CPU Core 수보다 크게 설정했을 경우에는 context-switch로 인한 부하가 발생할 수 있다.

HTH가 많아지면 여러 HTH를 관리하기 위한 부하가 발생하고, HTH 캐싱이 분산된다는 문제가 있을 수 있기 때문에 동시 사용자가 많지 않은 경우 NODE 절의 HTH 항목은 기본값인 1을 사용할 것을 권장한다.

7.1.2. Thread 설정

HTH 프로세스는 다른 프로세스들과 달리 멀티 스레드로 이루어져 있다. 이 스레드 관리가 성능에 영향을 줄 수 있으므로 이를 적당히 조절할 필요가 있다.

HTH 프로세스는 스레드간 통신을 담당하는 하나의 Selector 스레드와 그외 작업 스레드로 구성되어 있다.

다음은 작업 스레드에 대한 설명이다.

- Worker Thread

  이 스레드는 주로 요청을 처리하기 위한 대부분의 작업을 처리하는 스레드이다. 주로 HTML 요청을 처리하기 위해 요청 파일을 읽어드리는 역할을 한다. 또한 SSL이 사용되어지는 경우 SSL Handshake, 암호화, 복호화, 작업을 한다. 또한 Compression을 사용하는 경우 압축하는 작업을 한다. 그외 HTTP Authentication 작업등 대부분의 작업을 처리한다. Worker Thread 설정은 무조건 많이 설정하는 것보다 CPU 수에 맞춰 설정하는 것이 좋다. Worker Thread는 하나의 요청에 매이지 않고 맡겨진 작업(파일 읽기 등)만 처리하고 Selector에게 반환하므로 하나의 요청에 블록되지 않는다.

- Sendfile Thread

  특정 OS에서 지원하는 sendfile 기능을 사용할 수 있을 때 적용할 수 있는 스레드이다. 주로 요청 파일의 크기가 큰 경우 사용될 수 있으며, 크기가 큰 파일을 블록킹으로 처리하게 된다. 서비스하기 위한 파일 중 파일 크기가 큰 경우를 고려해서 설정해야 한다.

- AccessLog Thread

  HTH에서 모든 HTTP 요청이 끝나고 나면 AccessLog를 남겨야 하는데 이를 위한 스레드이다. 이 스레드를 사용하지 않으면 Selector 스레드가 WSM에 AccessLog를 남겨달라도 보내야 한다. 기본값을 사용하는 것이 좋다.

스레드가 많아지면 여러 스레드를 관리하기 위한 부하가 발생하고, 스레드간 context-switch로 인한 부하가 발생할 수 있으므로 동시 사용자가 많지 않은 경우 기본값을 사용할 것을 권장한다. 참고로 각 작업 스레드는 HTH 프로세스 하나당 생성되므로 전체 작업 스레드가 많아지지 않도록 주의해야 한다.

7.1.3. 예제

SSL 사용 빈도가 높아지면서, 대부분의 요청을 SSL로 처리하는 경우가 있을 수 있다. SSL 처리는 CPU 사용률이 높기 때문에 CPU를 효율적으로 사용하는 것이 중요하다. WebtoB는 이에 대해 프로세스 및 스레드를 CPU 환경에 맞춰 설정할 수 있다. 다음은 간단한 예이다.

아래와 같은 서버 환경에서 SSL 사용 빈도가 높은 경우를 예로 들 수 있다.

- 서버 OS : IBM AIX
- 물리 CPU 수 : 8
- 하나의 물리 CPU당 SMT-THREADING THREAD 수 : 4
- 가상 CPU 수 : 32

위와 같은 서버에서 운영자가 HTH 프로세스를 물리 CPU 수에 맞춰 설정하고, 작업 스레드 수를 물리 CPU당 SMT-THREADING THREAD 수에 맞춰 설정한다면 가장 최적인 환경을 구성할 수 있을 것이다.

WebtoB 설정 파일에서 HTH 프로세스의 수 및 HTH의 스레드를 설정하는 방법은 다음과 같다.

```
*DOMAIN
webtob1

*NODE
webmain
    WebtoBDir = "$WEBTOBDIR",
    SHMKEY = 69000,
    Docroot = "docs/",
    AppDir = "ap/",
    Port = "5469",
    HTH = 8,
    IndexName = "index.html",
    Logging = "accesslog",
    ErrorLog = "errorlog"

*HTH_THREAD
hworker
    WorkerThreads = 4
```

위와 같이 설정하면 WebtoB가 8개의 HTH 프로세스를 기동하게 된다. 각각의 HTH 프로세스는 4개의 작업 스레드를 사용하게 되고, 총 작업 스레드는32개가 된다. 즉, 서버의 물리 CPU, 가상 CPU에 맞춰 프로세스 및 스레드를 설정함으로써 효과적인 성능을 낼 수 있게 된다.

참고로 각각의 HTH 프로세스는 각 프로세스에 걸리는 로드에 따라서 적당히 사용자의 요구를 분배하기 때문에 하나의 HTH 프로세스가 특별히 일을 많이 하는 등의 문제는 발생하지 않는다.

7.2. 서버 프로세스 설정

다른 웹 서버의 경우 하나의 프로세스에 HTML, CGI, SSI 등의 모든 서비스 처리 루틴이 포함되어 있다. 따라서 사용자가 접속하는 경우 하나의 프로세스에서 모든 처리가 끝난다. 사용자당 담당 프로세스가 연결되는 구조에서는 각 사용자의 요구 순간마다 프로세스가 접속을 연결받아 처리한다. 이런 구조에서는 하나의 프로세스에서 모든 사용자의 Request를 처리할 수 있다는 장점은 있으나, 사용자가 사용하지도 않는 기능들이 의미없이 설정되어 있을 수도 있다. WebtoB는 이에 대해 필요한 서비스들을 분리시켜 제공한다. 다음은 간단한 예이다.

7.2.1. 예제

어느 쇼핑몰 사이트에서 100%의 사용자를 기준으로 다음과 같은 가정을 세운다.

- 60%가 HTML 이용자
- 20%가 Servlet 이용자
- 10%가 CGI 이용자
- 10%가 SSI 이용자

대부분의 경우 하나 혹은 2개 정도의 서비스에 사람이 몰리는 경우가 일반적이다.

문제점

다른 웹 서버의 경우 운영자는 100개의 프로세스를 설정하여 서비스할 것이다. 이는 사용자가 동시에 100명 정도가 들어오는 경우 반드시 필요한 수치이다. 위와 같은 사용자 분포도를 보인다면, 100개의 프로세스에서 60개는 거의 HTML만을 서비스하고 20개는 Servlet을 서비스하고, 10개가 CGI를 서비스하고 10개가 SSI를 서비스하게 된다. 각각의 프로세스에 포함된 다른 기능들은 의미없이 메모리만 차지한다. 게다가 사용자가 HTML만을 반복적으로 이용하는 경우 이런 현상은 더 더욱 두드러지게 되어 메모리를 효율적으로 운영할 수 없다.

해결 방법

이런 경우에 WebtoB는 위와 같은 문제를 다음과 같은 방법으로 해결한다.

WebtoB의 경우 특정 서비스가 분리되어 있기 때문에 각각의 서비스 프로세스 수를 조정할 수 있다. 가장 많은 HTML은 HTH의 Worker 스레드에서 처리하고, 그외 각각의 CGI, SSI 등의 서비스들을 별도의 독립적인 프로세스로 설정한다. 하나의 프로세스에서 모든 처리를 하는 것이 아니라, 각각의 서비스를 처리하는 것을 분리하여 별도로 각각의 수치를 조정한다.

위와 같은 경우 운영자가 CGI를 사용자가 많이 사용한다는 것을 알게 된다면, CGI를 처리하는 프로세스들을 많이 설정한다. 그리고 다른 프로세스들은 적게 설정하여 불필요한 메모리의 낭비를 막을 수 있다. HTH의 Worker 스레드를 6개, Servlet에 관련된 프로세스를 2개, CGI와 SSI와 관련된 프로세스를 1개씩 띄운다면 가장 최적인 환경에서 서비스를 할 수 있을 것이다.

이러한 설정은 WebtoB 설정 파일의 **SERVER 절**에 **'MinProc'**와 **'MaxProc'** 항목에서 한다. 자세한 사항은 SERVER 절을 참고한다.

다음은 SERVER 절의 설정에 대한 예이다.

```
*SERVER
jsv1        SvgName = jsvg, MinProc = 2, MaxProc = 10
cgi         SvgName = cgig, MinProc = 1, MaxProc = 10
ssi         SvgName = ssig, MinProc = 1, MaxProc = 2
```

항목	설명
MinProc	MinProc는 처음 WebtoB가 기동될 때 시작되는 프로세스의 수이다. 따라서 위의 경우 각 초기값은 다음과 같이 설정한다. ◦ Servlet(jsv1) : 2개 ◦ CGI(cgi) : 1개 ◦ SSI(ssi) : 1개
MaxProc	MaxProc는 최대로 이용할 수 있는 프로세스의 수이다. 이는 동적으로 WebtoB를 조정하는 것이 가능하기 때문에 필요한 값이다.

사용자가 초기값으로 설정한 후에 약간의 변화가 생겨 프로세스의 조정이 필요한 경우 MinProc와 MaxProc의 범위 안에서 조정하는 것이 가능하다. MinProc의 값은 MaxProc의 값보다 작거나 같아야 한다. **wsadmin**이라는 관리자용 툴에서 설정한다.

7.3. BRUN 발생하는 경우 튜닝 설정

다음과 같이 wsadmin 환경에서 st -p 명령의 수행을 통해 서버 프로세스의 정보를 확인할 수 있다.

```
$ wsadmin> st -p 정보

HTH 0(12689): RDY
   ------------------------------------------------------------------------
   svr_name   svgname    spr_no(pid)  status     count    avg(rt)  clid svc v
   ------------------------------------------------------------------------
   cgis       cgig         0( 12690)  BRUN         0   0.0000( 6)    0 cgi 0
```

위와 같이 조회된 정보 중 상태(status)가 'BRUN'인 경우가 있다. BRUN(Blocked Run) 상태는 서버 프로세스가 HTH에게 응답을 보내지 않고 잠시 멈춰있는 상태를 의미한다. 일반적으로 클라이언트가 응답을 받는 속도가 느린 경우에 발생한다. 크기가 큰 응답을 처리하는 경우 또는 모바일 환경과 같이 네트워크 속도가 느린 경우에 발생할 수 있다.

BRUN이 발생하면 서버 프로세스가 Block 상태이기 때문에 사용자는 서비스에 장애가 있다고 판단할 수 있지만, BRUN이 발생한 상황은 서비스의 장애가 아니다. 또한, BRUN이 자주 발생하면 다른 요청에 대한 큐잉(Queuing) 현상이 발생할 가능성이 높아진다.

일종의 서비스 지연 현상이라고 할 수 있는 BRUN 발생에 대한 튜닝 포인트는 **HttpOutBufSize** 및 **FlowControl** 설정이다. 이 2가지 설정을 통해 BRUN의 발생 빈도를 줄일 수 있다.

HTML을 처리하는 경우에는 BRUN이 발생하지 않는다. Worker Thread에서 HTML 요청을 처리하기 위해 파일을 읽게 되는데 ReadBufSize 단위로 읽어서 Selector Thread에게 전달후 Selector Thread에서 FlowControl하면 보내기 때문에 Worker Thread에 블록이 걸리는 일도 발생하지 않는다.

7.3.1. HttpOutBufSize 및 FlowControl 설정

다음은 BRUN의 발생 빈도를 줄이기 위해 HttpOutBufSize와 FlowControl을 설정한 예이다.

```
*SERVER
cgis        SvgName = cgig, MinProc = 5, MaxProc = 10,
            HttpOutbufSize = 4096, FlowControl = 50
```

- **HttpOutBufSize**

  HttpOutbufSize(기본값: 8192bytes)는 서버 프로세스가 응답을 만들어 HTH에게 전달할 때 버퍼의 단위이다.

  예를 들어 위와 같은 설정에서는 사이즈가 1024KB인 응답을 CGIS가 처리해야 하는 경우에 CGIS는 HttpOutbufSize(4KB)만큼 256(1024KB / 4KB)번 나눠서 이(조각)를 HTH에게 전달한다. 이는 한 번에 모두 처리하는 것이 아니라 4KB 버퍼에 담아서 HTH에게 전달하고, 다음 데이터를 동일한 4KB 버퍼에 담아서 HTH에게 전달하는 방식이다. 따라서 HttpOutbufSize 설정은 서버 프로세스가 HTH에게 전달하는 사이즈를 설정함과 동시에 응답 사이즈가 큰 경우에는 몇 번(몇 조각)씩 나눠서 전달할 것인지를 설정하는 것도 포함된다.

  참고로, JSV타입의 서버인 경우에는 응답을 전달하는 기준이 JSV 서버이기 때문에 JSV 서버에서 설정해야 하며,

HTH는 FlowControl 버퍼를 만드는데만 해당 값을 사용하게 된다.

- **FlowControl**

  HTH는 서버 프로세스로부터 전달되는 응답을 FlowControl 버퍼에 받아서 클라이언트에게 전달하게 되는데 FlowControl 버퍼의 크기는 (HttpOutBufSize * FlowControl)이다. 즉, FlowControl 설정은 HttpOutBufSize와 함께 FlowControl 버퍼의 크기를 조절하는 설정이다. (기본값: 50)

  클라이언트가 응답을 받는 속도가 느리다면 FlowControl 버퍼가 꽉 차게 되는 경우가 발생하고, 이때 더 이상 서버 프로세스로부터 전달되는 응답을 받을 수 없게 되므로 HTH는 HTMLS로부터 읽는 것을 중지한다. 이때 HTMLS는 BRUN이 발생하여 일시정지 상태가 된다. 클라이언트가 응답을 받아가면서 HTH의 FlowControl 버퍼가 절반 이상 비워지게 되면 HTMLS의 상태가 BRUN에서 RUN으로 전환되고, HTH는 HTMLS로부터 응답을 다시 읽기 시작한다.

HTH의 FlowControl 버퍼 크기를 크게 설정하는 것은 HTH가 사용하는 메모리 양 또한 증가한다는 것을 의미한다. 예를 들어 HTH는 한 요청에 대해 FlowControl 버퍼 크기(default 200KB = 4k * 50)만큼 사용하게 되고, 요청이 많아지는 경우 메모리 사용량은 요청 수만큼 증가하게 된다.

7.3.2. BRUN 상태를 줄이는 방법

BRUN 상태를 줄이기 위해서는 FlowControl 및 HttpOutBufSize 설정을 통해 FlowControl 버퍼의 크기를 조절하는 것이 필요하다.

다음은 1MB의 파일이 많은 경우의 HttpOutbufSize 및 FlowControl의 설정 변경 예제이다.

```
*SERVER
cgis        SvgName = cgig, MinProc = 5, MaxProc = 10,
            HttpOutbufSize = 8192, FlowControl = 100
```

HttpOutBufSize 및 FlowControl 설정을 크게 하면 HTH가 사용하게 되는 최대 메모리 양이 증가할 수 있다는 것에 주의한다. 특히, 4.1.5 이전 버전에서는 HttpOutBufSize 설정은 HTH에서 캐시하게 되는 파일 사이즈를 설정하므로, HttpOutBufSize 설정을 크게하면 HTH에서 캐시로 사용할 메모리 양도 증가하게 된다.

8. WebtoB 보안

WebtoB는 기본적으로 인증(Authentication)과 SSL(Secure Socket Layer)을 지원한다. 이는 기존의 다른 웹 서버들에서도 기본적으로 지원하는 것으로 웹에서 보안를 보장하기 위해 가장 많이 사용되는 방법들이다. 근본적으로 개방형 통신망을 지향하는 인터넷에서 사용자들의 정보를 보호하고, 자료의 유출을 막기 위해 전자 상거래 등의 거대한 규모의 사이트에서 반드시 검증하고 이용해야 할 방법들이다.

8.1. 인증 방법

인증(Authentication)의 방법은 매우 단순하다. 사용자가 사용자명과 비밀번호를 웹 서버에 보내고 웹 서버는 사용자가 접근 권한을 가지고 있는지 확인하기 위하여 이름과 암호화된 비밀번호가 있는 파일을 열고 해당 정보들을 검색한다. 그리고 검색된 정보에서 사용자명과 비밀번호의 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. 전체 모든 사용자들에 대한 설정도 가능하다. 실제 브라우저와 서버 간의 작동은 다음과 같다.

사용자가 어떤 서버에 접속해서 어떤 자료를 요청한다고 하는 경우 사용자가 요청한 자료가 중요한 자료이기 때문에 특정 사용자들만 보거나 실행할 수 있는 권한이 있다면, 서버는 사용자에게 "Authentication Required [HTTP 401]" 신호를 전달한다. 그러면 사용자는 웹 서버에 자신의 사용자명과 비밀번호를 같이 보내고 웹 서버는 이를 가지고 사용자 인증을 수행한다. 이때, 사용자의 사용자명과 비밀번호가 인터넷으로 전달되는데 정보가 유출되면 문제가 될 수 있기 때문에 사용자의 사용자명과 비밀번호를 암호화한다.

8.2. SSL

WebtoB에서는 전자 상거래 사이트를 위해서 안전한 보안 방식인 SSL(Secure Socket Layer)을 제공한다. WebtoB에서 구현된 SSL을 사용하는 방법을 설명하기 전에 먼저 SSL에 대해서 설명한다. SSL은 웹 서버가 제공하는 것이기 때문에 SSL에 대한 내용만 이해한다면 사용하는 방법은 아주 간단하다.

8.2.1. SSL v3.0

Netscape사에서 개발한 암호화 전송 프로토콜 (RC2, RC4로 암호화) X.509 Certificate를 이용하여 서버와 클라이언트간의 인증을 한다. 40bit Key를 사용하는 SSL은 보안에 있어서 신뢰를 주지 못하고, 128bit Key의 SSL을 사용하여야 하며 현재 SSL 버전은 3.0이다.

SSL은 Layer를 기반으로 하는 프로토콜이다. 각 Layer의 메시지는 length, description, content field를 포함한다. SSL이 전송해야 할 메시지를 가질 때 SSL은 그것을 제어 가능한 크기의 Block으로 나누고, 선택적으로 데이터를 압축한다. 압축한 데이터에 인증 코드(MAC : Message Authentication Code)을 추가하고 암호화를 한 후 결과를 전송한다. 데이터가 도착하게 되면 복호화를 하고 MAC을 인증한 후 압축된 데이터를 원래 상태로 만들고, Block을 재배열하여 얻어진 메시지를 상위 레벨로 넘겨주게 된다.

SSL을 지원하는 웹 서버에 연결을 할 경우 흔히 사용하는 URL 표기 방식인 "http://*" 대신에 **"https://*"**를 사용해야 한다. 보통의 경우(http)에 포트 번호 80을 사용하는데 SSL을 사용할 경우(https)에는 포트 번호 443을 호출하므로 하나의 브라우저를 이용하여 "http://*"와 "https://*"를 동시에 사용할 수가 있다.

SSL이 동작하기 위해서 필요한 사전 작업들이 "handshake" 과정에서 수행된다.

이 작업들을 정리하면 다음과 같다.

- 클라이언트와 서버가 서로 자신의 인증서(Certificate)를 교환한다. 그리고 각각은 자신이 받은 인증서에 기록되어 있는 유효기간, 서명을 확인한다.
- 클라이언트는 이후에 수행될 암호화와 메시지 인증 코드(MAC)의 생성에 필요한 난수(비밀 Key)를 생성하여 이것을 서버의 공개 Key로 암호화하여 서버에게 전송한다.
- 서비스를 받는 동안 사용할 암호화 알고리즘과 hash 함수의 종류를 결정한다. 이 과정에서는 클라이언트는 자신이 받아 들일 수 있는 암호화 알고리즘의 리스트를 서버에게 제시하고 이들 중에서 하나를 서버가 선택한다.

8.2.2. SSL vs. SHTTP

SHTTP는 EIT(Enterprise Integration Technology)에서 개발한 프로토콜로서 기존의 HTTP 프로토콜에 보안기능을 갖게 하기 위하여 몇 가지의 헤더를 추가한 것이다. SSL은 서비스를 단위로 암호화와 인증을 받지만 SHTTP의 경우에는 전달되는 메시지를 단위로 암호화와 인증을 받는다.

SSL과 비슷한 점이 있다면 클라이언트가 서버에게 SHTTP를 이용한 커넥션을 만드는 과정에서 다양한 종류의 암호화 기법들 중에서 하나를 선택할 수 있고 암호화의 정도를 선택할 수 있다는 점이다.

SHTTP가 지원하는 암호화 기법의 특징 및 알고리즘을 정리하면 다음과 같다.

- RSA, Diffie-Hellman 또는 Kerberos 인증기법을 사용한다.
- 공개 Key 인증서(certificate) 양식은 X.509 또는 PKCS-6를 사용한다.
- 디지털 서명 알고리즘은 RSA 또는 DSA(Digital Signature Algorithm)를 사용한다.

8.2.3. SSL Encryption

암호화를 40bit, 128bit 중 정하는 것은 주어진 데이터를 인코딩하는 데 필요한 Key의 길이이고, Key의 길이는 길수록 훨씬 안전하다. 보통 SSL 등을 지원하는 타 웹 서버들은 자체적으로 128bit를 지원하나, 이러한 128bit Encryption을 지원하기 위해서는 서버뿐만 아니라 클라이언트 역시 이를 지원해야 한다.

8.2.4. Ciphers

Ciphers는 암호화에 사용되는 알고리즘으로 더욱 더 안전하고 강력한 것들을 선택할 수 있다. 일반적으로 Ciphers가 암호화할 때 더 많은 bit를 사용할수록 그 데이터를 해독하기가 더 어렵고 어떠한 양방향 암호화 과정에서도 양자는 같은 Cipher를 사용해야 한다. 이러한 Cipher에는 여러 가지 종류가 있어서 서버는 가장 많이 알려진 것을 사용할 수 있어야 하고, 클라이언트가 서버와의 SSL 접속을 시작하면 정보를 암호화하는 데 선호하는 Cipher를 서버에 알리게 된다.

현재 WBSSL에서 제공되는 Ciphers 종류는 다음과 같다.

```
Cipher suite                  Protocols KeyExch.   Authen. Encryption    Mac
TLS_AES_256_GCM_SHA384        TLSv1.3   any        any     AESGCM(256)   AEAD
TLS_CHACHA20_POLY1305_SHA256  TLSv1.3   any        any CHACHA20/POLY1305(256) AEAD
TLS_AES_128_GCM_SHA256        TLSv1.3   any        any     AESGCM(128)   AEAD
ECDHE-RSA-AES256-GCM-SHA384   TLSv1.2   ECDH       RSA     AESGCM(256)   AEAD
```

```
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2   ECDH       ECDSA   AESGCM(256)   AEAD
ECDHE-RSA-AES256-SHA384       TLSv1.2   ECDH       RSA     AES(256)      SHA384
ECDHE-ECDSA-AES256-SHA384     TLSv1.2   ECDH       ECDSA   AES(256)      SHA384
ECDHE-RSA-AES256-SHA          SSLv3     ECDH       RSA     AES(256)      SHA1
ECDHE-ECDSA-AES256-SHA        SSLv3     ECDH       ECDSA   AES(256)      SHA1
SRP-DSS-AES-256-CBC-SHA       SSLv3     SRP        DSS     AES(256)      SHA1
SRP-RSA-AES-256-CBC-SHA       SSLv3     SRP        RSA     AES(256)      SHA1
SRP-AES-256-CBC-SHA           SSLv3     SRP        SRP     AES(256)      SHA1
DH-DSS-AES256-GCM-SHA384      TLSv1.2   DH/DSS     DH      AESGCM(256)   AEAD
DHE-DSS-AES256-GCM-SHA384     TLSv1.2   DH         DSS     AESGCM(256)   AEAD
DH-RSA-AES256-GCM-SHA384      TLSv1.2   DH/RSA     DH      AESGCM(256)   AEAD
DHE-RSA-AES256-GCM-SHA384     TLSv1.2   DH         RSA     AESGCM(256)   AEAD
DHE-RSA-AES256-SHA256         TLSv1.2   DH         RSA     AES(256)      SHA256
DHE-DSS-AES256-SHA256         TLSv1.2   DH         DSS     AES(256)      SHA256
DH-RSA-AES256-SHA256          TLSv1.2   DH/RSA     DH      AES(256)      SHA256
DH-DSS-AES256-SHA256          TLSv1.2   DH/DSS     DH      AES(256)      SHA256
DHE-RSA-AES256-SHA            SSLv3     DH         RSA     AES(256)      SHA1
DHE-DSS-AES256-SHA            SSLv3     DH         DSS     AES(256)      SHA1
DH-RSA-AES256-SHA             SSLv3     DH/RSA     DH      AES(256)      SHA1
DH-DSS-AES256-SHA             SSLv3     DH/DSS     DH      AES(256)      SHA1
DHE-RSA-CAMELLIA256-SHA       SSLv3     DH         RSA     Camellia(256) SHA1
DHE-DSS-CAMELLIA256-SHA       SSLv3     DH         DSS     Camellia(256) SHA1
DH-RSA-CAMELLIA256-SHA        SSLv3     DH/RSA     DH      Camellia(256) SHA1
DH-DSS-CAMELLIA256-SHA        SSLv3     DH/DSS     DH      Camellia(256) SHA1
GOST2001-GOST89-GOST89        SSLv3     GOST       GOST01  GOST89(256)   GOST89
GOST94-GOST89-GOST89          SSLv3     GOST       GOST94  GOST89(256)   GOST89
ECDH-RSA-AES256-GCM-SHA384    TLSv1.2   ECDH/RSA   ECDH    AESGCM(256)   AEAD
ECDH-ECDSA-AES256-GCM-SHA384  TLSv1.2   ECDH/ECDSA ECDH    AESGCM(256)   AEAD
ECDH-RSA-AES256-SHA384        TLSv1.2   ECDH/RSA   ECDH    AES(256)      SHA384
ECDH-ECDSA-AES256-SHA384      TLSv1.2   ECDH/ECDSA ECDH    AES(256)      SHA384
ECDH-RSA-AES256-SHA           SSLv3     ECDH/RSA   ECDH    AES(256)      SHA1
ECDH-ECDSA-AES256-SHA         SSLv3     ECDH/ECDSA ECDH    AES(256)      SHA1
AES256-GCM-SHA384             TLSv1.2   RSA        RSA     AESGCM(256)   AEAD
AES256-SHA256                 TLSv1.2   RSA        RSA     AES(256)      SHA256
AES256-SHA                    SSLv3     RSA        RSA     AES(256)      SHA1
CAMELLIA256-SHA               SSLv3     RSA        RSA     Camellia(256) SHA1
PSK-AES256-CBC-SHA            SSLv3     PSK        PSK     AES(256)      SHA1
ECDHE-RSA-AES128-GCM-SHA256   TLSv1.2   ECDH       RSA     AESGCM(128)   AEAD
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2   ECDH       ECDSA   AESGCM(128)   AEAD
ECDHE-RSA-AES128-SHA256       TLSv1.2   ECDH       RSA     AES(128)      SHA256
ECDHE-ECDSA-AES128-SHA256     TLSv1.2   ECDH       ECDSA   AES(128)      SHA256
ECDHE-RSA-AES128-SHA          SSLv3     ECDH       RSA     AES(128)      SHA1
ECDHE-ECDSA-AES128-SHA        SSLv3     ECDH       ECDSA   AES(128)      SHA1
SRP-DSS-AES-128-CBC-SHA       SSLv3     SRP        DSS     AES(128)      SHA1
SRP-RSA-AES-128-CBC-SHA       SSLv3     SRP        RSA     AES(128)      SHA1
SRP-AES-128-CBC-SHA           SSLv3     SRP        SRP     AES(128)      SHA1
DH-DSS-AES128-GCM-SHA256      TLSv1.2   DH/DSS     DH      AESGCM(128)   AEAD
DHE-DSS-AES128-GCM-SHA256     TLSv1.2   DH         DSS     AESGCM(128)   AEAD
DH-RSA-AES128-GCM-SHA256      TLSv1.2   DH/RSA     DH      AESGCM(128)   AEAD
DHE-RSA-AES128-GCM-SHA256     TLSv1.2   DH         RSA     AESGCM(128)   AEAD
DHE-RSA-AES128-SHA256         TLSv1.2   DH         RSA     AES(128)      SHA256
DHE-DSS-AES128-SHA256         TLSv1.2   DH         DSS     AES(128)      SHA256
DH-RSA-AES128-SHA256          TLSv1.2   DH/RSA     DH      AES(128)      SHA256
DH-DSS-AES128-SHA256          TLSv1.2   DH/DSS     DH      AES(128)      SHA256
DHE-RSA-AES128-SHA            SSLv3     DH         RSA     AES(128)      SHA1
DHE-DSS-AES128-SHA            SSLv3     DH         DSS     AES(128)      SHA1
DH-RSA-AES128-SHA             SSLv3     DH/RSA     DH      AES(128)      SHA1
DH-DSS-AES128-SHA             SSLv3     DH/DSS     DH      AES(128)      SHA1
DHE-RSA-SEED-SHA              SSLv3     DH         RSA     SEED(128)     SHA1
```

```
DHE-DSS-SEED-SHA              SSLv3    DH          DSS     SEED(128)     SHA1
DH-RSA-SEED-SHA               SSLv3    DH/RSA      DH      SEED(128)     SHA1
DH-DSS-SEED-SHA               SSLv3    DH/DSS      DH      SEED(128)     SHA1
DHE-RSA-CAMELLIA128-SHA       SSLv3    DH          RSA     Camellia(128) SHA1
DHE-DSS-CAMELLIA128-SHA       SSLv3    DH          DSS     Camellia(128) SHA1
DH-RSA-CAMELLIA128-SHA        SSLv3    DH/RSA      DH      Camellia(128) SHA1
DH-DSS-CAMELLIA128-SHA        SSLv3    DH/DSS      DH      Camellia(128) SHA1
ECDH-RSA-AES128-GCM-SHA256    TLSv1.2  ECDH/RSA    ECDH    AESGCM(128)   AEAD
ECDH-ECDSA-AES128-GCM-SHA256  TLSv1.2  ECDH/ECDSA  ECDH    AESGCM(128)   AEAD
ECDH-RSA-AES128-SHA256        TLSv1.2  ECDH/RSA    ECDH    AES(128)      SHA256
ECDH-ECDSA-AES128-SHA256      TLSv1.2  ECDH/ECDSA  ECDH    AES(128)      SHA256
ECDH-RSA-AES128-SHA           SSLv3    ECDH/RSA    ECDH    AES(128)      SHA1
ECDH-ECDSA-AES128-SHA         SSLv3    ECDH/ECDSA  ECDH    AES(128)      SHA1
AES128-GCM-SHA256             TLSv1.2  RSA         RSA     AESGCM(128)   AEAD
AES128-SHA256                 TLSv1.2  RSA         RSA     AES(128)      SHA256
AES128-SHA                    SSLv3    RSA         RSA     AES(128)      SHA1
SEED-SHA                      SSLv3    RSA         RSA     SEED(128)     SHA1
CAMELLIA128-SHA               SSLv3    RSA         RSA     Camellia(128) SHA1
IDEA-CBC-SHA                  SSLv3    RSA         RSA     IDEA(128)     SHA1
PSK-AES128-CBC-SHA            SSLv3    PSK         PSK     AES(128)      SHA1
ECDHE-RSA-RC4-SHA             SSLv3    ECDH        RSA     RC4(128)      SHA1
ECDHE-ECDSA-RC4-SHA           SSLv3    ECDH        ECDSA   RC4(128)      SHA1
ECDH-RSA-RC4-SHA              SSLv3    ECDH/RSA    ECDH    RC4(128)      SHA1
ECDH-ECDSA-RC4-SHA            SSLv3    ECDH/ECDSA  ECDH    RC4(128)      SHA1
RC4-SHA                       SSLv3    RSA         RSA     RC4(128)      SHA1
RC4-MD5                       SSLv3    RSA         RSA     RC4(128)      MD5
PSK-RC4-SHA                   SSLv3    PSK         PSK     RC4(128)      SHA1
ECDHE-RSA-DES-CBC3-SHA        SSLv3    ECDH        RSA     3DES(168)     SHA1
ECDHE-ECDSA-DES-CBC3-SHA      SSLv3    ECDH        ECDSA   3DES(168)     SHA1
SRP-DSS-3DES-EDE-CBC-SHA      SSLv3    SRP         DSS     3DES(168)     SHA1
SRP-RSA-3DES-EDE-CBC-SHA      SSLv3    SRP         RSA     3DES(168)     SHA1
SRP-3DES-EDE-CBC-SHA          SSLv3    SRP         SRP     3DES(168)     SHA1
EDH-RSA-DES-CBC3-SHA          SSLv3    DH          RSA     3DES(168)     SHA1
EDH-DSS-DES-CBC3-SHA          SSLv3    DH          DSS     3DES(168)     SHA1
DH-RSA-DES-CBC3-SHA           SSLv3    DH/RSA      DH      3DES(168)     SHA1
DH-DSS-DES-CBC3-SHA           SSLv3    DH/DSS      DH      3DES(168)     SHA1
ECDH-RSA-DES-CBC3-SHA         SSLv3    ECDH/RSA    ECDH    3DES(168)     SHA1
ECDH-ECDSA-DES-CBC3-SHA       SSLv3    ECDH/ECDSA  ECDH    3DES(168)     SHA1
DES-CBC3-SHA                  SSLv3    RSA         RSA     3DES(168)     SHA1
PSK-3DES-EDE-CBC-SHA          SSLv3    PSK         PSK     3DES(168)     SHA1
```

8.3. 인증 서비스

8.3.1. 인증기관

인터넷과 같은 개방형 통신망에서 일어날 수 있는 사용자 데이터의 보안 및 사용자 신원인증을 위한 방법으로는 공개 Key 암호화 알고리즘이 있다. 이 알고리즘은 데이터를 암호화하는 Key와 암호화된 문을 해독하는 Key가 쌍으로 구성된다.

비밀 Key는 해독할 사람 본인이 가지고 있고, 공개 Key는 공개하여 다른 사람들이 비밀 Key의 소유자에게 메시지를 전할 때 그 공개 Key로 암호화해서 보내는 방식이다. 이 방법에서 문제가 되는 것은 그 공개 Key가 정말로 사용자 본인의 것인가 아닌가 하는 점인데, 이를 확인하기 위해 공개 Key를 증명해주는 기관을 별도로 두게 되며, 이를 인증기관(Certification Authority, CA)이라고 한다.

메시지 송신자와 수신자가 모두 신뢰할 수 있는 제3의 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), 웹 서버 인증서, 사용자 인증서의 3종류로 구분한다.

8.3.2. 인증서

인증서(Certificate)란 어떤 소프트웨어나 어떤 기관 또는 개인을 보증하는 Q마크와 같은 것이다. 잘 모르는 신제품에 대하여 고객은 KS나 Q마크를 믿고 그 제품을 구입하게 되는데 인터넷에서도 이와 같은 개념으로 도입된 것이 인증서이다. 반대로 상점의 입장에서 보면 물건을 사러 온 고객이 믿어도 좋은 사람인지 고객의 신분증을 보고 신용카드 거래를 허용할 수도 있다.

인증서는 사용하는 입장에서 보면 '서버 인증서'와 '클라이언트 인증서' 2가지로 나누어 볼 수 있고 트리(Tree) 구조를 따라서 보다 상위 기관에서 그 신뢰도를 인증하는 계층구조로 이루어져 있는 것이 보통이다. 사용자 데이터 보호와 사용자 신원의 추가 확인용으로만 사용될 뿐 전자 서명용으로 사용할 때는 전자서명법에 의한 법적효력을 갖지 않는 것이 일반적이다.

온라인 인증서는 인증하고자 하는 대상에 관한 몇 가지 내용을 기술한 문서라고 볼 수 있는데 개인적으로 생성할 수도 있고 발급기관에 요청할 수 있으며 그 형식은 ITU(International Telecommunication Union)에서 제정한 X.509의 양식 또는 RSA Data Security사에서 제정한 PKCS-6의 양식 및 PEM을 따르는 인증서를 사용한다. 이러한 인증서는 cert.crt와 같은 형식으로 배포될 수 있으며 이것은 개인 Key를 제외하고 배포되는 공개 Key라고 생각할 수 있다.

다음은 PKCS-6의 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 인증서의 각 의미는 다음과 같다.

①은 인증서의 시리얼 번호이다.

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

③은 인증서를 발급한 기관에 관한 정보를 나타낸다.

구분	설명
C	country
O	Organization
OU	Organization Unit

④는 인증서의 유효기간이 '1994년 11월 9일 15시 54분17초'에서부터 '1999년 12월 31일 15시 54분 17초'까지라고 표시하고 있다. CA는 인증서를 발급해주는 기관의 서버를 의미한다.

8.3.3. 인증서 발급 정책

확실한 보안 통신을 위해서는 서버뿐만 아니라 클라이언트에서의 인증서가 필요하다. 클라이언트의 경우에 "정말로 내가 요청한 서비스를 해줄 적법한 서버인가?"를 확인하는 것이 필요할 것이고, 서버의 경우에 "정말로 서비스를 요청한 실체가 적법한 클라이언트인가?"를 확인하는 작업이다. 보통 서버는 의무적으로 인증서를 제공하고, 클라이언트에서 인증서를 요구할 것인가 하는 것은 발급하는 정책에 따라 여러 가지가 있을 수 있다.

인증서 발급 정책(Certificate Issuing Policy)은 일반적으로 다음의 4가지 형태로 존재하게 된다.

- 인증과정이 필요 없다. (No certificate required.)
- 클라이언트는 올바른 인증서를 제공할 수도 있다. 인증서가 제공되면, 서버가 가지고 있는 인증서와 일치하는 인증기관(Certification Authority)에서 인증한 것이어야 한다.
- 클라이언트는 올바른 인증서를 반드시 제공해야 한다.
- 클라이언트는 올바른 인증서를 제공할 수도 있다. 그러나 반드시 서버의 인증기관과 일치할 필요는 없다.

일반적으로, 영화표 예매나 사이버 증권사와 같은 곳에서는 고객의 ID와 비밀번호만 있으면 서버의 인증서를 바탕으로 보안 통신이 이루어지게 되는 type0 형식을 사용한다. 한국통신에서 진행하는 banktown 21c 프로젝트에서는 보다 확실한 보안을 위한 type1 형식을 사용한다. 서비스를 이용하기 위해서는 웹 사이트에서 **[인증서 제출]** 버튼을 클릭해서 인증서를 제출하면(일반적으로는 브라우저에서 이것을 제공하게 할 수도 있다) 전자통장이 실행되고, 동시 서버에서 인증서를 발급받아 상호 간의 인증(Mutual Authentication)을 획득하게 되는 것이다. type0과 같이 단순히 비밀번호를 입력하는 방법보다는 인증서를 이용하는 방법이 인증서와 비밀번호를 동시에 이용하기 때문에 더욱 안전하다.

은행의 ATM 단말기를 이용할 때 은행에서 발급한 카드를 넣고 비밀번호를 입력한 후에 현금을 인출한다. 이러한 과정에서 카드가 인증서에 해당한다고 볼 수 있다. ATM 단말기에 넣은 카드로 은행의 서버에 자신의 실체를 알리고,

비밀번호 입력을 통해 적법한 사용자임을 인증하는 것이다. 이것은 디지털 서명에서 사용한 방법과 동일한 것이다.

8.3.4. Verisign에서 서버 인증서 발급 받기

Verisign은 세계적으로 가장 유명한 사설 인증기관이다. Verisign 홈페이지(www.verisign.com)에서 SSL Server Certificate를 선택하면 인증서를 받을 수 있다. 인증서는 구입하여 영구적으로 권한을 사용할 수도 있고, 일정기간(14일) trial로 사용해 볼 수도 있는데, 실제로 구입하기 위해서는 약간 더 복잡한 과정이 필요하다.

인증서란 어떤 대상을 인증기관에서 신용을 보증하는 것이기 때문에 Verisign사는 인증서를 요청한 기관이나 업체에 사업허가서 등의 제반 서류를 요청하게 되는데, D&B라는 Business Database에 등록되어 있다면 이러한 과정이 훨씬 간단해 질 수 있고, 현재는 주로 미국 내의 기업들만이 등록되어 있다.

이전에 설명외에 SSL은 더 복잡한 구조와 인증 방법을 가지고 있다. 또한, SSL(Secure Socket Layer)은 TCP 프로토콜과 HTTP 프로토콜 레벨 사이에 있는 데이터를 암호화할 수 있는 Layer이기 때문에, SSL의 암호화를 통하여 데이터가 인터넷 등의 개방형 통신망으로 나간다 할지라도 암호를 해독할 가능성이 거의 없어 데이터에 대한 보안성이 생기게 된다. 따라서 이는 보안성이 특히 중요한 전자상거래 등에서 많이 사용되며, 국내에서도 대부분의 전자상거래에서 많은 사용자들이 이용하고 있다.

하지만 이는 TCP 프로토콜과 HTTP 프로토콜 레벨 사이에서 데이터를 암호화해야 하기 때문에 성능상에 문제점을 초래할 수 있다. 특히 SSL을 지원하는 타 웹 서버에서는 SSL Package와 복잡한 연동 문제로 인하여 처리 속도가 느려지는 문제점이 발생하게 된다. 그 이유는 웹 서버가 SSL을 외부의 함수로 이용하여 자신들의 내부 엔진과 결합성이 떨어지기 때문이다. WebtoB에서는 이들을 내부 엔진과 결합하여, 성능에 문제가 될 수 있는 모든 요인들을 제거하고 효율성에 초점을 맞추었다.

실제 SSL의 사용은 매우 단순하다. 현재 사용자들이 많이 이용하는 브라우저가 거의 모두 SSL을 지원하기 때문에 https로 시작하는 사용자 요구들을 스스로 SSL로 간주하여 웹 서버로 보내게 된다. 또한, 웹 서버에서는 이런 요구를 받아 위에서 설명한 인증 방법 등을 동원하여 사용자를 인증하게 된다. 이는 모두 웹 서버와 사용자 브라우저 내에서 이루어지는 일로 사용자가 복잡하게 직접 관여할 필요는 없다.

8.4. 인증과 SSL의 사용

본 절에서는 인증에 사용되는 명령어와 SSL 사용법에 대해서 설명한다.

8.4.1. 인증

wsmkpw

WebtoB에서 인증을 설정하는 방법은 환경설정에 **AUTHEN 절**에서 설명한다. 자세한 내용은 AUTHENT 절의 설명을 참고한다.

인증을 이용하기 위해서는 WebtoB에서의 설정뿐 아니라 실제 사용자의 사용자명과 비밀번호를 만들어서 저장해야 할 필요가 있다. 사용자명과 비밀번호를 생성하기 위해서 **wsmkpw** 실행 파일을 사용한다. wsmkpw 파일은 WebtoB가 설치된 디렉터리에 있는 "bin/" 디렉터리 아래에 있다.

다음은 wsmkpw의 사용법과 옵션에 대한 설명이다.

```
$ wsmkpw [-d] [-p passwd] [-r realm] <file_path> <username>
```

- 항목

항목	설명
file_path	인증 파일이 있는 전체 경로이다.
user_name	인증에 사용될 사용자 이름이다.

- 옵션

옵션	설명
[-d]	Digest로 인증하기 위한 암호를 생성한다.
[-p passwd]	옵션을 통해 암호를 입력한다. 이 옵션을 사용하면 암호를 입력받는 프롬프트 없이 암호를 생성한다. 터미널에 암호가 노출되므로 주의해야 한다.
[-r realm]	Digest 인증을 위한 realm을 설정한다.

다음은 "/data1/gloria/webtob/auth" 아래에 passwd라는 파일이 생기고 newuser라는 새로운 사용자가 저장되고 내부에는 입력한 비밀번호가 암호화되어 저장되는 예제이다. 이 파일을 이용하여 WebtoB는 실제 인증을 수행한다.

```
wsmkpw /data1/gloria/webtob/auth/passwd newuser
New Password: *****
Retype Password: *****
```

인증(Authentication)의 설정

WebtoB의 환경설정 중에 **AUTHENT 절**에 UserFile 항목에 passwd라는 파일을 등록하면 WebtoB가 인증을 수행하는 과정에서 이 파일을 이용하여 인증 작업을 하게 된다.

wsmkpw 실행 파일을 이용하여 사용자명과 비밀번호를 등록한 후 원하는 항목에 이것을 추가하면 사용자가 그 항목에 대한 요구를 하는 경우마다 WebtoB는 인증 작업을 수행한다.

인증 작업은 간단한 응용으로 사용자의 인증 작업을 수행할 수 있기 때문에 웹의 초기에 많이 이용되었다. 그러나 인증에 사용하는 암호화 방법이 아주 일반적이고도 해독이 가능하기 때문에 전자 상거래를 구현한 사이트 등에서는 많이 외면 되고 있다. 또한 사용자의 비밀번호가 인터넷에 유출되면 나쁜 의도로 비밀번호를 중간에 capture하여 해독하는 사례도 발생하면서 더 강력한 인증과 암호화 방법이 강구되었다.

이러한 문제에 대처하기 위해 Netscape사에 의해서 **SSL**(Secure Socket Layer)이 등장하였다. 이는 기존 암호화 방법을 한 차원 높게 만든 것으로 비밀번호와 사용자명을 가져갔다 하더라도 암호를 해독하는 것이 아주 어렵고 거의 불가능하기 때문에 보안에 완벽을 기할 수 있는 방법으로 인정받고 있다.

WebtoB에서 인증과 SSL을 이용하기 위해서는 WebtoB의 환경 파일에 이를 이용하기 위한 별도의 설정이 필요하다.

인증 기능을 사용하기 위해서는 **AUTHENT 절**을 설정해야 한다. AUTHENT 절은 WebtoB에서 보안을 위하여

설정할 인증에 대한 정의를 하는 절이다. 그리고 AUTHENT 절에서 선언한 것을 각각의 서버 그룹에 선언을 한다. 각 서버 그룹 단위로 인증을 설정하는 것이 가능하다. 만약 사용자가 CGI에 대해서 인증을 사용하는 경우 CGI를 선언한 서버 그룹에 AUTHENT 절에서 선언한 인증 이름을 선언한다. 자세한 내용은 설정 항목을 참고한다.

다음은 AUTHENT 절을 설정하는 방법이다.

```
*AUTHENT
AUTHENT name        Type,
                    UserFile
```

항목	설명
AUTHENT name	사용자가 원하는 이름을 설정한다.
Type	사용자가 이용하고 싶은 Encryption 방법을 설정한다. 일반적으로 이용되는 방법은 Basic과 Digest, MD5, RSA 등이 있다. WebtoB에서는 Basic과 Digest를 지원한다.
UserFile	wsmkpw를 이용하여 생성된 password 파일을 설정한다. password 파일을 이 AUTHENT 절에서 이용해야 하기 때문에 그 파일의 위치를 입력해야 한다. password 파일은 사용 전에 생성되어야 한다.

다음은 AUTHENT 절을 설정한 예제이다.

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

위와 같은 설정을 하였다면 사용자가 원하는 설정으로 인증 파일을 사용할 기본적인 준비가 된 것이다. 해당 설정을 사용해서 원하는 작업에 이 AUTHENT 절에서 정의한 항목을 실제로 연결하면 된다. 사용자가 이용하는 WebtoB의 설정 항목은 'AUTHENT name' 이다. 이를 이용하여 AUTHENT 절에 선언한 인증 방법을 이용한다. 설정에 대한 자세한 내용은 AUTHENT 절을 참고한다.

간단한 예로 사용자가 CGI 작업에 인증을 설정하고 싶다면 다음과 같이 설정한다(authen1은 위의 예에서 설정되었다).

```
*SVRGROUP
cgig      NodeName = webmain, SvrType = CGI,
          AuthentName = authent1
```

위와 같은 형식으로 설정하면 WebtoB는 사용자가 CGI 서비스를 요청할 때마다 사용자에게 인증 작업을 수행할 것이다. 이때 주의할 것은 이 인증 작업이 서버 그룹 단위로 이루어진다는 점이다.

따라서 CGI 그룹을 설정하면 이에 대응하는 서버들은 모두 인증 작업을 수행할 것이다. 따라서 만약 사용자가 특정 CGI는 인증을 수행하고 또 다른 CGI는 아니라면 2개의 서버 그룹을 설정해야 한다.

이에 대한 예는 다음과 같다.

```
*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 = "${WEBTOBDIR}/cgi-bin/"
alias2        Uri = "/cgi/",
              Realpath = "${WEBTOBDIR}/cgi/"
```

위와 같이 2가지로 그룹을 설정한 후 인증을 원하는 것은 cgig_authent라는 서버 그룹을 이용하여 서버를 설정하고, 다른 것들은 cgig를 그대로 이용하여 설정하면 된다. 이런 설정은 SVRGROUP 절에 선언될 수 있는 모든 것에 적용된다.

8.4.2. SSL 설정

WebtoB에서 SSL을 이용하기 위해서는 별도의 설정이 필요하다. 다른 인증이나 LOG 절의 선언과 마찬가지로, SSL 절을 선언하고 이를 이용하는 방식으로 진행된다. 자세한 내용은 SSL 절을 참고한다.

WebtoB에서 SSL을 이용하기 위해서는 먼저 NODE 절이나 SSL을 이용하고 싶은 Virtual Host를 선언한 VHOST 절에서 SslFlag 항목을 설정해야 한다. 기본적으로 WebtoB는 SSL을 사용하도록 되어 있지 않다. SslFlag 항목은 'Y'나 'N'으로 설정하는 것이 가능하다. 'Y'라는 값을 입력하면, SSL을 사용한다는 것이 되고, 'N'은 이와 반대의 역할을 한다. SslFlag 항목에 대한 설정이 없으면, 기본적으로 SSL을 이용하는 것이 아니기 때문에, 'N'은 거의 사용하지 않는다.

다음은 SslFlag 항목의 사용 예이다.

```
SslFlag = Y
```

위와 같이 SslFlag를 'Y'로 설정하여 SSL을 사용하겠다고 설정한 후에는 SSL에 대한 설정을 해야 한다.

다음은 SSL 절의 형식이다.

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

위 예제는 기본적인 설정에 필요한 정보만 기술한 것이다. 설정한 SSL을 AUTHENT 절에서 하듯이 각각의 SSL을 설정한다. SSL 설정은 SslName 항목을 사용한다. 설정하는 SslName 항목의 내용은 SSL 절에서 설정한 내용과 동일하게 설정한다.

인증관련 내용은 SVRGROUP 절에서 설정하고 SSL 절은 NODE 절이나 VHOST 절에서 SSL을 사용하는 곳에 설정한다.

다음은 SslName 항목 설정에 대한 예제로 ssl1으로 설정된 값으로 SSL을 사용하는 예제이다.

```
SslName = "ssl1"
```

ssl1이라는 항목은 SSL 절에서 설정한다. 다음은 SSL 절 설정에 대한 예제이다.

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

다음은 SSL을 실제 적용하는 방법에 대한 예제이다. NODE 절에서 SSL을 이용하기 위해서 SslFlag 항목을 설정하고 SslName 항목을 설정한다.

```
*NODE
test        WebtoBDir = "/user/webtob",
            SHMKEY = 69000,
            HTH = 1,
            Port = "80",
            SslFlag = Y,
            SslName = "ssl1",
          ...
```

SslName = "ssl1" 설정에 의해 이 NODE는 ssl1에서 정의된 항목을 통해서 SSL 서비스를 한다는 것을 알 수 있다.

9. WBAPI

WebtoB는 기존 CGI 방식의 애플리케이션 프로그램들의 문제점을 해결하기 위해 새로운 WebtoB만의 API를 제공한다. 본 장에서는 WebtoB에서 제공하는 API에 대해서 설명한다.

9.1. 개요

기존의 CGI 방식은 사용자의 요구가 발생하는 경우마다 새로운 프로세스가 시작되어 사용자의 요구를 처리한다. 사용자가 새로운 요구를 하면 그 순간에 시스템의 운영체제가 그 프로그램에 해당하는 프로세스를 Fork하는 작업을 수행하게 한다. 새로운 프로세스를 Fork하는 작업은 시스템 자체에 부하를 걸게 되고, 이런 일이 많아지게 되면 시스템에 많은 부담을 준다. 이런 문제점 때문에 기존 CGI 사용자들은 점차 새로운 애플리케이션으로 관심을 돌리게 되었고, Servlet, PHP, ASP, JSP 등이 등장하였다.

이런 프로그램은 모두 C로 만들어진 프로그램이 아니라 Java나 스크립트 등을 이용한 새로운 형태의 프로그램들이다. 따라서 새로운 개발에는 용이할지 몰라도 기존 C Style로 CGI 프로그램들을 개발하여 운영해 온 사람들에게는 기존 프로그램들을 모두 폐기하고 새로 만들어야 하는 부담을 안겨주고 있다.

CGI 개발자들은 자신들이 운영하던 CGI를 버리고 완전히 새로운 프로그램을 도입할 것인지, 아니면 기존 프로그램을 운영하되 서버를 늘려서 CGI의 부담을 줄일 것인지 고민한다. WebtoB에서는 기존 CGI 개발자들의 고민을 해결해주기 위해 새로운 형태의 API를 개발하였다.

제공되는 API는 C Style의 프로그램을 위한 API로서 CGI 프로그램의 부담을 덜고 기존 CGI 프로그램의 수정은 최소화할 수 있게 한다. 이를 통해 기존 CGI 프로그램으로 서비스를 해 오던 많은 사용자들은 부담도 덜고, 성능도 향상시킨다.

9.2. 개념

CGI 방식은 위에서 언급한 대로 사용자가 요구할 때마다 매 순간 새로운 프로그램이 Fork되고 처리가 끝나는 순간 Exit되는 일을 반복한다. 이는 다른 처리를 해야 할 시간에 새로운 프로세스를 메모리에 로드하고 스케줄링을 해야 하는 등의 많은 작업을 해야 하기 때문에 시스템의 관점에서는 굉장히 큰 부담이다.

다음 그림은 Apache 웹 서버에서의 CGI를 사용하는 모습을 나타낸다.

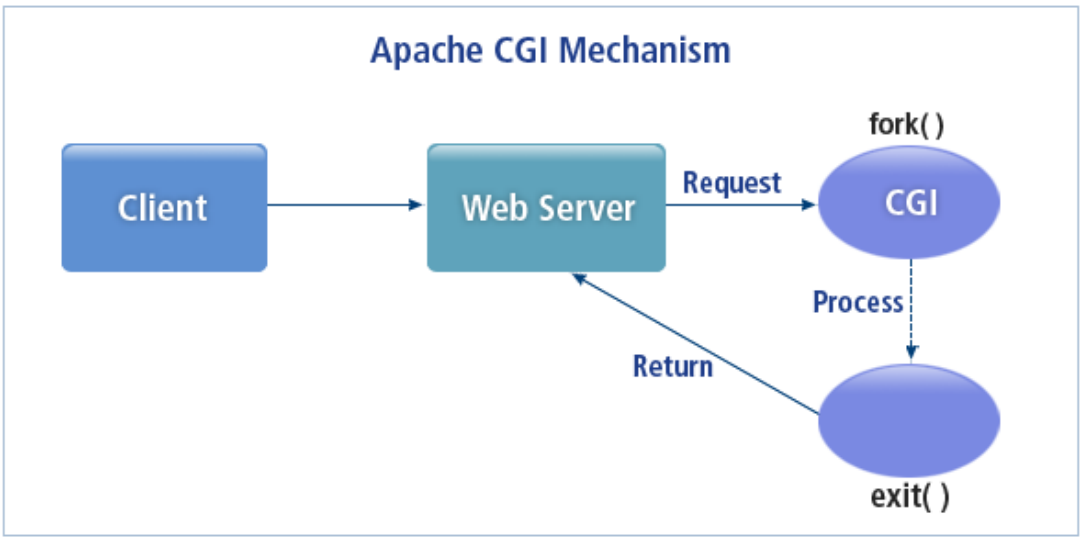


그림 5. Apache에서의 CGI 동작 방식

위 그림을 보면 웹 서버가 Request를 요청할 때마다 CGI가 Fork되는 로드가 생길 수 밖에 없다.

WebtoB에서 제공하는 WBAPI는 우선 CGI의 이런 점을 개선하였다. 즉, 사용자가 요구를 보내는 매 순간에 프로세스를 Fork하는 것이 아니라, 웹 서버가 Boot Up되는 시점에 이러한 프로그램들을 미리 Fork하여 메모리에 올려 놓는 것이다. 또한, 사용자의 요구에 대한 처리가 끝난 시점에서도 Exit 되는 것이 아니라, 계속 메모리에 상주하여 다음 요구를 기다리는 것이다. 결국 매 순간 프로세스가 Fork되고 Exit되는 부담이 줄어들게 되는 것이다.

또한 단순히 이러한 부담을 줄이는 외에도 이 WBAPI로 설계된 프로그램의 관리를 WebtoB가 담당하게 된다. 따라서 만약 사용자의 부주의나 설계 오류로 프로그램이 다운되거나 문제가 생겼을 때 즉각 새로운 프로그램을 다시 실행할 수 있게 하는 등의 기능이 추가되었다. WebtoB는 프로그램들의 수를 조정해서 특정 프로그램에 대한 요구가 많다고 판단되면 이를 미리 많이 실행시켜서 부하를 줄인다.

WebtoB의 WBAPI의 호출을 그림으로 나타내면 다음과 같다.

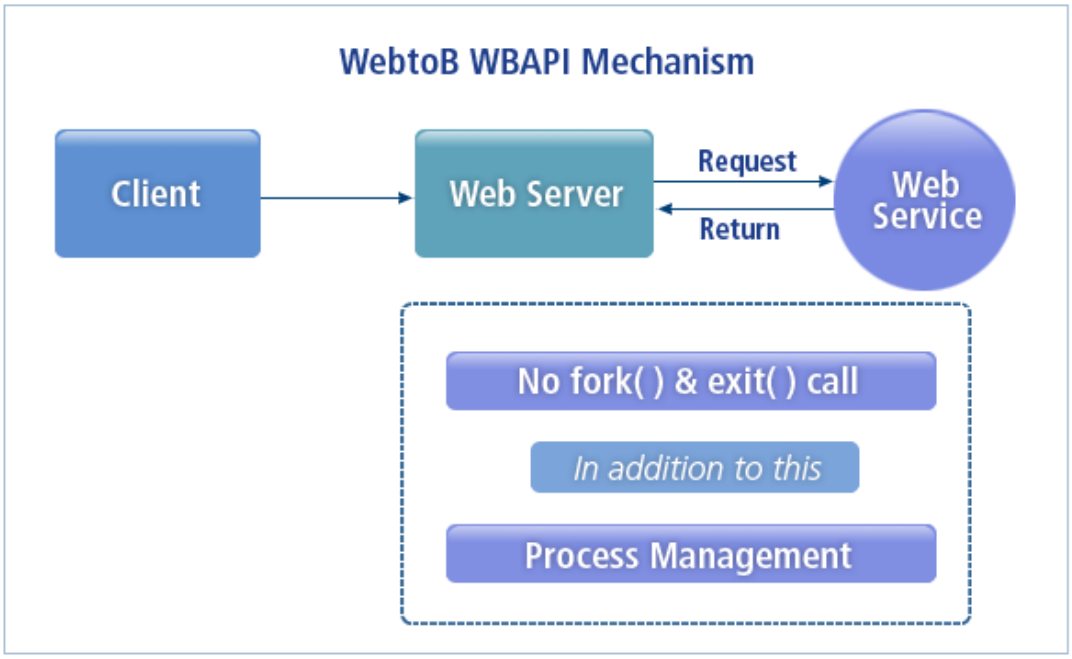


그림 6. WebtoB에서의 WBAPI

WBAPI는 main()으로 시작하는 일반 함수와는 다르다. 이는 WBAPI로 작성한 프로그램들이 독립된 프로세스로

발생하기는 하지만, 실제 WebtoB의 내부와 연결된 프로세스로 동작하기 때문이다. 따라서 함수의 시작이 일반적인 main()이 아닌 Service Name이라는 것으로 시작하게 된다. 따라서 함수의 시작되는 이름이 반드시 Service Name이 되어야 하고, 이는 WebtoB의 환경 파일에 등록이 되어 있어야 한다. 이에 대한 자세한 내용은 이후에 설명될 환경설정을 참고한다.

9.3. 환경설정

WebtoB에서 WBAPI를 사용하기 위해서는 프로그램의 작업과 함께 WebtoB의 환경설정이 필요하다. 즉, WebtoB에 WBAPI를 이용하여 프로그램을 개발하였고, 이는 이렇게 이용하겠다는 식의 등록을 한다.

다음은 WebtoB의 환경 파일에 대한 예이다. 아래 설정 파일에서 진한색으로 처리된 부분이 실제 WBAPI를 사용하기 위한 기본 설정이다.

```
*DOMAIN
webtob1

*NODE
webmain
    WebtoBDir = "/usr/local/webtob",
    SHMKEY = 99000,
    Docroot = "/usr/local/webtob/docs",
    AppDir = "/usr/local/webtob/ap",
    Hostname = "webmain.tmax.co.kr",
    Port = "8080",
    JsvPort = 9099

*SVRGROUP
cgig    NodeName = webmain, SvrType = CGI
jsvg    NodeName = webmain, SvrType = JSV
ssig    NodeName = webmain, SvrType = SSI
wbapg  NodeName = webmain, SvrType = WEBSTD

*SERVER
cgi     SvgName = cgig, MinProc = 10, MaxProc = 10
ssi     SvgName = ssig, MinProc = 2, MaxProc =2
jsv1    SvgName = jsvg, MinProc = 10, MAXProc = 10
webaps SvgName = wbapg, MinProc = 5, MAXProc = 5

*SERVICE
write_board  SvrName = webaps

*URI
uri1  Uri ="/svct/", SvrType = WEBSTD, SvrName = webaps
```

- SVRGROUP 절

  webmain라는 이름의 Host 머신에 wbapg라는 서버 그룹을 설정하고, 이것이 WBAPI를 사용하기 위한 WEBSTD라는 서버 유형으로 설정한다.

  ```
  wbapg        NodeName = webmain, SvrType = WEBSTD
  ```

- SERVER 절

SVRGROUP 절에 설정된 wbapg에 있는 것이 webaps라는 서버로 정의되어 있음을 나타낸다.

```
webaps       SvgName = wbapg, MinProc = 5, MaxProc = 5
```

SVRGROUP 절, SERVER 절이 항상 설정되어야 할 기본이다. 이는 대부분의 환경에서 Host 머신의 이름만 다를 뿐 별 차이가 없다. 실제 서비스를 응용하기 위해서는 각 서비스에 대한 설정이 추가로 필요하게 된다.

- SERVICE 절

  WBAPI로 개발된 프로그램의 등록을 한다. 실제 프로그램의 함수명과도 같은데 WBAPI로 개발된 프로그램이 일반 C Style과 달리 write_board()로 시작한다. 즉, 이 함수 시작 이름이 서비스의 이름이 되는 것이다. 따라서 등록할 때 이 이름으로 등록을 한다. SERVICE 절은 SERVER 절 다음에 추가한다.

```
*SERVICE
write_board    SvrName = webaps
```

- URI 절

  WBAPI로 만든 서비스를 사용할 때 URI 절을 설정한다. 요청에 특정 URI, 즉 "/svct/" 라는 URI로 시작하는 경우 WBAPI 서비스로서 실행시키겠다는 의미이다.

```
*URI
uri1        Uri =”/svct/”,
            SvrType = WEBSTD,
            SvrName = webaps
```

9.4. 서비스 테이블 생성

보통 WebtoB를 기동하기 위해서는 **wscfl**을 통해서 컴파일만 하면 되지만 WBAPI를 사용하기 위해서는 wscfl로 컴파일한 후 서비스 테이블을 생성해야 한다. bin/ 디렉터리 안에 있는 **wsgst**라는 명령어를 사용해서 생성한다. wsgst를 수행하면 WebtoB는 WBAPI를 쓰기 위한 '_svctab.c'라는 확장자를 가진 서비스 테이블을 지정된 경로에 생성한다.

wsgst

다음은 wsgst 명령을 사용해서 서비스 테이블을 사용하는 방법과 옵션에 대한 설명이다.

- 사용법

```
$ wsgst [-f 이진 WebtoB 환경파일 이름]
```

옵션	설명
[-f 이진 *WebtoB* 환경파일이름]	서비스 테이블을 생성할 때 참조하는 환경 파일이 wsconfig 아닌 경우 이 이름을 지정한다.

- 예제

  다음은 wsgst을 사용하는 예제이다.

```
$ wsgst   (Default로 wsconfig를 참조한다)
$ wsgst -f wsconfig2   (wscfl -o를 사용한 경우)
```

이로써 WBAPI를 사용하기 위한 모든 준비가 완료되었다. 이제 WBAPI로 개발된 프로그램을 컴파일하여 실행 파일을 생성하고 이를 위와 같은 형식으로 등록한다.

9.5. 프로그램의 컴파일

WBAPI로 개발된 프로그램은 C에서 기본적으로 제공되는 라이브러리 외에도 WebtoB에서 제공하는 특수한 라이브러리 및 서비스 테이블 파일을 이용하기 때문에 이를 컴파일할 때 연결해 주어야 한다.

아래는 UNIX/Linux에서 사용하는 컴파일을 위한 Makefile 파일의 예제이다.

```
#Makefile.common
TARGET= webaps
APOBJS= ap.o

#hp 32bit
CFLAGS = -Ae +DA1.1 +DD32 -O -I$(WEBTOBDIR)
#hp 64bit
#CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(WEBTOBDIR)
#sun 32bit
#CFLAGS = -O -I$(WEBTOBDIR)
#sun 64bit
#CFLAGS = -xarch=v9 -O -I$(WEBTOBDIR)
#Linux
#CFLAGS = -O -I$(WEBTOBDIR)
#ibm 32bit
#CFLAGS = -q32 -O -I$(WEBTOBDIR) -brtl
#ibm 64bit
#CFLAGS = -q64 -O -I$(WEBTOBDIR) -brtl

WEBTOB_INCDIR = $(WEBTOBDIR)/usrinc
WEBTOB_BINDIR = $(WEBTOBDIR)/bin
WEBTOB_LIBDIR = $(WEBTOBDIR)/lib
OBJS    = $(APOBJS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

.SUFFIXES : .v
.c.o:
        $(CC) $(CFLAGS) -c $<

wbaps: $(APOBJS) svct
        $(CC) $(CFLAGS) -L$(WEBTOB_LIBDIR) -o $(TARGET) $(OBJS) $(LIBS) $(USERLIBS)

svct:
        cp $(WEBTOBDIR)/svct/$(TARGET)_svctab.c .
        $(CC) $(CFLAGS) -I$(WEBTOB_INCDIR) –c $(TARGET)_svctab.c
```

다음은 Windows의 경우이다.

```
#Makefile for Windows
TARGET = webaps.exe

LIBS    = /link $(WEBTOBDIR)\lib\aps.lib ws2_32.lib
CFLAGS  = /D _DEBUG /I $(WEBTOBDIR) /Gd /MD /nologo

APOBJ   = $(TARGET:.exe=.obj)
SVCTSRC = $(TARGET:.exe=_svctab.c)
SVCTOBJ = $(TARGET:.exe=_svctab.obj)
OBJS    = $(APOBJ) $(SVCTOBJ)

$(TARGET): svct $(OBJS)
        cl $(CFLAGS) -o $@ $(OBJS) $(LIBS)
        copy $@ $(WEBTOBDIR)\ap

svct:
        copy $(WEBTOBDIR)\svct\$(SVCTSRC)

clean:
        -del $(OBJS) $(TARGET)
```

9.6. WBAPI의 시작 및 응용

작업이 완료되면 WebtoB가 기동될 때 WBAPI로 개발된 프로그램을 같이 실행시켜 프로세스 형태로 메모리에 로드된다.

CGI를 변환하는 것은 WBAPI의 기능 중의 일부이다. 기존 CGI의 변환 외에도 C Style에 능숙한 사람들은 새로운 애플리케이션을 개발할 때 이 WBAPI를 이용하면 개발의 편의성 및 성능 향상에도 많은 도움이 될 것이다. WebtoB는 지속적으로 WBAPI를 제공하고 또 향상할 것이므로 차후의 확장성이나 지원등에서 무리가 없을 것이다.

그리고 이 WBAPI를 통해서 기존 TP-Monitor에서 제공한 서비스 기능을 이용하는 것도 가능하다.

9.7. WBAPI의 종류

WBAPI는 각각 INIT/DONE, ALLOC, GET, PUT/SET, SEND, COOKIE, SESSION, ETC 계열로 구성되어 있다. WBAPI는 기능별로 나누어진 것으로 각각의 API에 대한 간단한 설명을 한다.

9.7.1. INIT/DONE API

INIT/DONE 계열의 함수는 개발자가 서비스를 시작하기 전의 초기화 과정이나 서비스 프로세스가 종료될 때 수행되어야 할 루틴을 수행한다.

다음은 INIT/DONE 계열 함수의 목록이다.

함수 이름	기능
wbSvrInit()	서버를 초기화한다.

함수 이름	기능
wbSvrDone()	WebtoB의 반환을 알리는 함수이다.

9.7.2. ALLOC API

ALLOC 계열의 함수는 사용자가 메모리를 할당하고 해제한다.

다음은 ALLOC 계열 함수의 목록이다.

함수 이름	기능
wbMalloc()	사용자가 설정한 SIZE만큼의 메모리를 할당한다.
wbFree()	wbMalloc()에 의해 할당된 메모리를 제거(free)한다.

9.7.3. GET API

GET 계열의 함수는 사용자가 요청한 것에 대하여 사용자의 정보를 읽어들이는 것으로 사용자의 Request의 데이터, 메소드 정보 값 등을 알아낸다.

다음은 GET 계열 함수의 목록이다.

함수 이름	기 능
wbGetAuthType()	사용자의 인증방식을 얻어낸다.
wbGetContentLength()	클라이언트가 보낸 Content의 크기를 얻어낸다.
wbGetDocumentRoot()	WebtoB의 Home 디렉터리의 값을 얻어낸다.
wbGetHdr()	Request의 헤더에서 특정 Key에 해당하는 값을 얻는다.
wbGetDateHdr()	데이터를 표현하는 헤더를 얻는다.
wbGetIntHdr()	지정된 헤더 값을 정수 형태로 얻는다.
wbGetNthHdr()	헤더에서 n번째에 해당하는 헤더 값을 얻는다.
wbGetHdrCount()	사용자가 보낸 Request의 헤더의 총 개수를 얻는다.
wbGetData()	Request의 데이터 필드에서 Key에 해당하는 값을 얻는다.
wbGetNthKey()	Request에서 n번째의 Key에 대한 데이터를 얻는다.
wbGetNthData()	Request의 데이터에서 n번째 데이터를 얻는다.
wbGetDataCount()	Request에서 입력으로 들어온 데이터의 숫자를 얻는다.
wbGetValue()	Request에서 특정 Key 값의 n번째 데이터를 얻는다.
wbKeyOccur()	Request로 들어온 데이터에서 Key 값의 수를 얻는다.
wbGetMethod()	Request로 들어온 HTTP 메소드를 정수형으로 읽어 들인다.
wbGetParsedURI()	클라이언트로부터 Request가 들어온 URI 정보를 얻어낸다.
wbGetPathInfo()	Request와 관계된 상대 경로 정보를 얻어낸다.
wbGetPathTranslated()	Request와 관계된 절대 경로 정보를 얻어낸다.

함수 이름	기 능
wbGetQueryString()	Request URL로부터 질의어 문자열을 얻어낸다.
wbGetProtocol()	Request와 관계된 프로토콜 정보를 얻어낸다.
wbGetRemoteUser()	Request하는 사용자의 이름을 얻어낸다.
wbGetRequestURI()	Request URI를 얻어낸다.
wbGetRemoteAddr()	Request한 Remote Host의 IP를 얻어낸다.
wbGetRemoteHost()	Request하는 Remote Host의 이름을 얻어낸다.
wbGetRemoteIdent()	서버로부터 검색된 사용자의 이름을 얻어내는 데 사용된다.
wbGetReqLine()	Request에 들어온 line의 첫 번째 line pointer를 얻는다.
wbGetFileName()	파일의 이름을 알아낸다.
wbGetFileLen()	파일의 크기를 알아낸다.
wbGetScheme()	Request된 서비스의 프로토콜 정보를 얻어낸다.
wbGetScriptFileName()	Request한 WBAPI 서비스가 실행되는 절대 경로를 얻어낸다.
wbGetScriptName()	Request한 서비스가 실행되는 경로를 얻어낸다.
wbGetServerName()	서버의 hostname을 얻어낸다.
wbGetServerPort()	서버의 포트 번호를 얻어낸다.
wbGetServerSoftware()	서버의 소프트웨어 정보를 얻어낸다.
wbGetTranslatedURI()	Request된 서비스의 URI를 해석하여 실제 경로를 얻어낸다.
wbGetRequestURL()	Request URL 값을 얻어낸다.

9.7.4. PUT/SET API

PUT/SET 계열의 함수는 사용자의 Request에 대한 응답으로 WBAPI에서 처리한 값을 출력하기 위한 용도로 쓰이는 것으로 CGI에 익숙한 사람이라면 출력값을 나타내기 위하여 C 언어의 **printf** 함수나 Perl의 **print 스크립트**를 이용하는 것과 같다고 보면 된다.

WBAPI에서는 print 스크립트에 대한 처리를 한 후 그 결과를 **WBSVCINFO**라는 구조체에 저장하여 그 구조체를 브라우저에 넘기는데 이 WBSVCINFO 구조체에 응답으로 보낼 데이터를 저장하는 역할을 하는 함수들이다.

PUT 계열 함수를 이용할 때는 몇 가지 주의 사항이 있는데 PUT 관련 함수들은 순서를 지켜주어야 WBAPI가 문제없이 수행된다.

- 다른 PUT 계열 함수들보다 먼저 수행되어야 하는 함수가 있다.

  wbSetStatus(WBSVCINFO *rqst, int status, char *status_msg) 함수는 Response 헤더를 먼저 작성하는 함수로 먼저 수행되어야 한다.

- PUT 관련 함수 중에서 헤더를 작성하는 함수들이 데이터 필드를 작성하는 함수들보다 먼저 선언되어야 한다.

다음은 PUT/SET 계열 함수의 목록이다.

함수 이름	기 능
wbPutHdr()	Response 헤더를 설정한다.
wbPutIntHdr()	Response 헤더를 추가한다.
wbSetStatus()	Request의 status 값을 설정한다.
wbPutStr()	string 값을 출력한다.
wbPut()	특정 사이즈만큼의 데이터를 출력한다.
wbPrint()	사용자가 지정한 내용을 출력한다.
wbPutFile()	File download를 위한 함수, 특정 파일을 읽어들여 모두 브라우저로 보내준다.
wbPutPartialFile()	File download를 위한 함수로 특정 파일의 '일부분'을 읽어서 브라우저로 보내준다. File의 offset부터 사이즈만큼 읽어서 들여 보내준다. 사이즈가 0이면 파일 끝까지 읽어 보내준다.

9.7.5. SEND API

SEND 계열은 사용자의 Request에 대한 응답으로 PUT 계열 함수를 사용하여 WBSVCINFO 구조체에 서비스 처리 결과를 저장한 것을 브라우저에 리턴하는 함수이다.

다음은 SEND 계열 함수의 목록이다.

함수 이름	기능
wbFlush()	현재 버퍼에 있는 모든 내용을 기록하게 한다.
wbSendError()	상태코드의 에러 여부를 확인한다.
wbSendRedirect()	지정된 주소로 응답을 되돌려준다.
wbReturn()	WebtoB의 반환을 알려준다.

9.7.6. COOKIE API

COOKIE 계열 함수는 HTTP 스타일의 Cookie를 생성하고, 읽고 처리하기 위한 작업을 수행한다.

다음은 COOKIE 계열 함수의 목록이다.

함수 이름	기 능
wbCreateCookie ()	새로운 Cookie를 생성한다.
wbGetCookie()	브라우저에 의해 보내진 Cookie 중에서 원하는 Cookie를 리턴한다.
wbPutCookie()	지정된 Cookie에 응답을 덧붙인다.
wbCookieGetDomain ()	지정된 Cookie에서 도메인을 리턴한다.
wbCookieGetName()	지정된 Cookie의 이름을 리턴한다.
wbCookieGetPath()	Cookie의 경로를 리턴한다.
wbCookieGetValue()	Cookie의 값을 리턴한다.
wbCookieGetVersion()	Cookie의 버전을 리턴한다.

함수 이름	기 능
wbCookieSetComment()	Cookie의 주석 필드를 설정한다.
wbCookieSetDomain()	Cookie의 도메인을 설정한다.
wbCookieSetMaxAge()	Cookie의 수명시간을 설정한다.
wbCookieSetPath()	Cookie를 위한 경로를 설정한다.
wbCookieSetSecure()	Cookie가 SSL과 같이 보호되는 프로토콜에서 전송되어야 하는지를 설정한다.
wbCookieSetValue()	Cookie에 새로운 값을 설정한다.
wbCookieSetVersion()	Cookie에 새로운 버전을 설정한다.

9.7.7. SESSION API

SESSION 계열 함수는 세션을 생성하고, 세션의 고유 정보를 얻어 기타 작업들을 수행할 수 있도록 한다.

다음은 SESSION 계열 함수의 목록이다.

함수 이름	기 능
wbGetRequestedSessionId()	사용자에 요청에 의해 명확해진 세션 ID를 얻어낸다.
wbGetSession()	생성된 세션이 없는 경우 세션을 생성하고 그 세션 값을 얻어낸다.
wbIsRequestedSessionIdValid()	요청된 세션이 올바른 세션이고 현재 사용 중이라면 true값을 얻어낸다.
wbSessionGetId()	세션에 할당된 유일한 ID를 얻는다.
wbSessionGetValueNames()	세션에 바인딩된 모든 객체의 이름을 포함하는 배열을 얻어낸다.
wbSessionGetCreationTime()	세션이 생성된 시간을 얻어낸다.
wbSessionGetValue()	세션에 바인딩된 객체의 값과 이름을 얻어낸다.
wbSessionSetValue()	세션에 지정된 객체의 값과 이름을 바인딩한다.
wbSessionIsNew()	세션이 새로운 것인지 아닌지 나타내는 값을 얻어낸다.
wbSessionRemoveValue()	지정된 이름에 바인딩된 객체를 제거한다.
WbSessionInvalidate()	세션을 무효화한다.
wbSessionGetMaxInactiveInterval()	세션에 설정된 세션 유지 시간을 얻어낸다.
wbSessionSetMaxInactiveInterval()	지정된 세션에 세션 유지 시간을 설정한다.
wbSessionGetLastAccessTime()	클라이언트가 마지막으로 요청을 보낸 시간을 얻어낸다.

9.7.8. ETC API

ETC 계열 함수는 WebtoB에 File Upload 수행하거나 WebtoB 에러 번호를 얻을 수 있다.

다음은 ETC 계열 함수의 목록이다.

함수 이름	기 능
wbGetErrno()	WebtoB의 에러 번호를 얻어낸다.
wbSaveFile()	File Upload를 위한 함수이다.

9.8. WBAPI를 이용한 CGI의 변환

WBAPI는 차후에 Multi Thread 기능을 제공할 예정이다. 이를 위하여 각각의 함수마다 특정 자료 구조(Data Structure)를 선언하는 것이 필요하다. 처음 WBAPI 프로그램을 작성하게 되면 Service Name을 등록하게 되는데 이때 **WBSVCINFO**라는 구조를 선언하게 되어 있다. 이를 WBAPI 내부의 각각의 함수의 앞에 반드시 입력해야 한다. 따라서 한 WBAPI 프로그램 내부의 모든 WBAPI 함수는 Service Name이 선언된 구조를 함수의 첫 번째 인자로 가지고 있어야 한다.

본 절에서는 WBAPI를 이용하여 기존의 CGI 프로그램을 변환할 때 어떤 방식으로 이루어지는지 설명한다.

9.8.1. CGI 프로그램

다음은 CGI 방식을 이용해서 간단한 게시판을 구현하는 프로그램의 예이다.

```
#include "qDecoder.h"

int strcheck(char *str) {
	if (str == NULL)
		return 0;
	if (strlen(str) == 0)
		return 0;
	return 1;
}

int main(void) {
	char *name, *title, *doc;
	char *email, *homepage;

	/* 입력값들을 얻어냄 */
	name = qValue("writer");
	title = qValue("title");
	doc = qValue("doc");
	email = qValue("email");
	homepage = qValue("homepage");

	/* 입력값들이 올바른지 검사 */
	if (!strcheck(name))
		qError("이름을 입력하세요.");
	if (!strcheck(title))
		qError("글 제목을 입력하세요.");
	if (!strcheck(doc))
		qError("본문을 입력하세요.");
	if (strcheck(email))
		if (!qCheckEmail(email))
			qError("E-Mail 주소를 정확히 입력하세요.");
	if (strcheck(homepage))
```

```
            if (!qCheckURL(homepage))
                    qError("Homepage URL을 정확히 입력하세요.");

      /* 처리결과출력 – 입력확인 */
      qContentType("text/html");
      printf("<HTML>\n\n");
      printf("<HEAD><TITLE>Bestbook 게시판 – 글 올리기 확인</TITLE></HEAD>\n\n");
      printf("<BODY>\n");
      printf("<H2> Bestbook 게시판 - 글 올리기 확인 </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("올리신 글은 다음과 같습니다.<BR><BR>\n");
            printf("<TABLE WIDTH=600 BORDER=1>\n");
            printf("<TR><TD><B>제목</B> : %s</TD></TR>\n", title);
            printf("<TR><TD>%s</TD></TR>\n", doc);
            printf("</TABLE>\n\n<BR>이 글이 정상적으로 게시판에 올려졌습니다.\n");
            printf("</BODY>\n\n</HTML>");
            qFree();
}
```

대부분의 CGI 프로그램의 구조는 간단하고 단순하다. QUERY_STRING을 통해서 전달된 사용자의 입력 값을 바탕으로 특정한 작업을 수행한 후 이를 HTML 문서와 같은 형태로 다시 전달한다. HTML 형태의 문서를 만드는 것은 주로 **printf** 함수를 통하여 이루어진다.

프로그램에 있는 여러 함수들은 거의 C 언어에서 기본적으로 제공하는 것이고, qValue나 strcheck 등의 처리 함수들은 입력변수의 값을 읽어 오거나, 변수 값이 맞는지 분석하는 함수들이다. 이 역시 간단한 라이브러리 형태로 지원된다.

위의 CGI 프로그램을 분석하면 다음과 같다.

1. CGI 프로그램에서는 writer, title, doc, email, homepage란 변수를 이용한다. 따라서 사용자는 이들 변수의 값을 전달한다. 아래의 프로그램은 사용자가 전달한 값을 CGI 프로그램에서 인식하기 위하여 분리하는 과정이다.

   각각의 변수에 qValue라는 함수를 이용하여 값을 읽어낸다.

   ```
   name = qValue("writer");
   title = qValue("title");
   doc = qValue("doc");
   email = qValue("email");
   homepage = qValue("homepage");
   ```

2. 각각의 변수값을 읽어낸 후 아래의 strcheck 함수를 통해서 각각의 변수 값이 모두 입력되었는지 확인한다.

   ```
   if (!strcheck(name))
             qError("이름을 입력하세요.");
   ```

```
if (!strcheck(title))
          qError("글 제목을 입력하세요.");

if (!strcheck(doc))
          qError("본문을 입력하세요.");

if (strcheck(email))
   if (!qCheckEmail(email))
       qError("E-Mail 주소를 정확히 입력하세요.");

if (strcheck(homepage))
    if (!qCheckURL(homepage))
         qError("Homepage URL을 정확히 입력하세요.");
```

3. 함수들을 모두 정상적으로 수행하였다면 CGI로 전달된 사용자의 변수 값이 모두 이상이 없다는 것이다. 이제 게시판에 값을 올리고 출력 형태를 만들어야 한다.

   아래의 함수는 이 CGI 프로그램이 제공하는 출력 형태가 HTML임을 나타내고 있다.

```
qContentType("text/html");
```

4. printf 함수를 통하여 결과 값을 만들어내는 작업을 수행한다. 이는 HTML 문서를 실제로 Editing하는 것과 거의 같다.

   다음을 보면 printf 함수를 통해서 HTML Tag를 만들어 낸다.

```
printf("<HTML>\n\n");
printf("<HEAD><TITLE>Bestbook 게시판 - 글올리기 확인</TITLE></HEAD>\n\n");
printf("<BODY>\n");
printf("<H2> Bestbook 게시판 - 글 올리기 확인</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("올리신 글은 다음과 같습니다.<BR><BR>\n");
printf("<TABLE WIDTH=600 BORDER=1>\n");
printf("<TR><TD><B>제목</B> : %s</TD></TR>\n", title);
printf("<TR><TD>%s</TD></TR>\n", doc);
printf("</TABLE>\n\n<BR>이 글이 정상적으로 게시판에 올려졌습니다.\n");
printf("</BODY>\n\n</HTML>");
```

9.8.2. WBAPI 프로그램

다음은 CGI 프로그램과 똑같은 기능을 하는 WBAPI 프로그램이다. 구조적인 것은 CGI 프로그램과 같다.

1. 각각의 변수에 대한 값을 읽어내야 한다.

   CGI 프로그램은 qValue라는 함수를 사용하는데, WBAPI에서는 **wbGetData**라는 함수를 사용한다. 아래의

함수들은 위의 CGI에서 qValue를 통해서 수행된 것을 그대로 수행하는 것이다. 결국 각각의 변수값을 읽어들이게 된다.

```
writer = wbGetData(rqst, "writer");
title = wbGetData(rqst, "title");
doc = wbGetData(rqst, "doc");
email = wbGetData(rqst, "email");
homepage = wbGetData(rqst, "homepage");
```

2. 변수를 읽어 들인 값을 확인하는 과정은 strcheck를 이용한다. 처리 방법은 CGI 프로그램과 동일하다.
3. WBAPI에서는 사용자의 환경변수 뿐만 아니라, 헤더에 전송된 값도 읽어 들일 수 있다. 이것은 아래와 같은 **wbGetHdr** 함수를 통하여 이루어진다.

   다음은 Content-Length 값을 읽어 들이는 방법이다.

   ```
   hd = wbGetHdr(rqst, "Content-Length");
   ```

4. 일련의 작업들이 끝났다면 이제 출력을 위한 문서를 만들어야 한다.

   CGI 프로그램에서는 printf 함수로 처리된 것이었다. WBAPI에서는 **wbPrint**라는 함수로 처리된다. 결국 기존의 CGI에서 printf 함수를 썼던 부분을 모두 wbPrint로 처리하면 된다. 결과 형태가 약간 달라질 수 있지만 이는 출력 형태인 HTML Tag가 약간 변화한 것일 뿐 게시판을 만든다는 것에는 변화가 없다.

   CGI 프로그램에서 printf를 통하여 만들어진 헤더 정보는 이곳에서는 **wbPutHdr**를 통해서 생성된다. 결국 HTML 안에 포함되는 것만 wbPrint를 이용하고 헤더 정보는 wbPutHdr를 이용한다.

   다음은 헤더를 생성하는 예제이다.

   ```
   wbPutHdr(“ContentType”, “text/html");
   ```

   다음은 HTML Tag를 생성하는 예제이다.

   ```
   wbPrint(rqst, "<HTML>\n\n");
   wbPrint(rqst, "<HEAD> <TITLE> webtob Write Board </TITLE> </HEAD>\n\n");
   wbPrint(rqst, "<BODY>\n");
   wbPrint(rqst, "<H2> Bestbook</H2>\n");
   wbPrint(rqst, "<H3>Writer is%s </H3> \n",writer);
   wbPrint(rqst, "<H3>Title is%s </H3> \n",title);
   wbPrint(rqst, "<HR>\n");
   wbPrint(rqst, "<H2> %s </H2> \n",doc);
   wbPrint(rqst, "<H3> newvalue :%s%d</H3> \n",hd,rt);
   wbPrint(rqst, "<HR ALIGN=left>\n<BR>");
   wbPrint(rqst, ".<BR><BR>\n");
   wbPrint(rqst, "<TABLE WIDTH=600 BORDER=1>\n");
   wbPrint(rqst, "</TABLE>\n\n<BR>.\n");
   wbPrint(rqst, "</BODY>\n\n</HTML>");
   ```

5. 위의 모든 작업이 끝나면 결과가 끝났다는 것을 마지막에 return 대신에 **wbReturn**을 수행하는 것으로 WebtoB에 전달한다.

```
wbReturn(rqst, WBSUCCESS);
```

괄호 안의 WBSUCCESS는 결과가 성공적으로 수행되었다는 것을 나타낸다.

Appendix A: 환경설정 파일 예제

본 부록에서는 기본 환경설정 파일과 JEUS 연동을 위한 환경설정 파일의 예제를 제공한다.

A.1. 기본 환경설정 파일

다음은 JEUS와 연동하지 않는 기본 WebtoB 환경설정 파일에 대한 예제이다.

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDIR = "$WEBTOBDIR",
  SHMKEY = 56000,
  Dcoroot = "docs/",
  Port = "8080",
  HTH = 1,
  Logging = "accesslog",
  ErrorLog = "errorlog",
  SysLog = "systemlog"

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
  NodeName = mynode,
  SvrType = CGI
ssig
  NodeName = mynode,
  SvrType = SSI

*SERVER
cgi
  SvgName = cgig,
  MinProc = 4,
  MaxProc = 10
ssi
  SvgName = ssig,
  MinProc = 2,
  MaxProc = 10

*URI
cgi_bin
  Uri = "/cgi-bin/",
  Svrtype = CGI

*EXT
htm
  Mimetype = "text/html",
  SvrType = HTML

*ALIAS
```

```
alias
  Uri = "/cgi-bin/",
  Realpath = "${WEBTOBDIR}/cgi-bin/"

*LOGGING
accesslog
  Format = "default",
  Filename = "log/access_%Y%%M%%D%.log"
errorlog
  Format = "ERROR",
  Filename = "log/error_%Y%%M%%D%.log"
systemlog
  Format = "SYSLOG",
  Filename = "log/system_%Y%%M%%D%.log"
```

A.2. WebtoB와 JEUS 연동 환경설정 파일

다음은 JEUS에 연결된 경우 WebtoB 환경설정 파일에 대한 예제이다.

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 84565,
  Docroot = "docs/",
  Port = "8080",
  HTH = 1,
  Logging = "accesslog",
  ErrorLog = "errorlog",
  SysLog = "systemlog",
  JsvPort = 9999

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
htmlg
  NodeName = mynode,
  SvrType = HTML
jsvg
  NodeName = mynode,
  SvrType = JSV

*SERVER
MyGroup
  SvgName = jsvg,
  Minproc = 10,
  MaxProc = 10

*URI
examples
  Uri = "/examples/",
  SvrType = JSV
```

```
*EXT
htm
  Mimetype = "text/html",
  SvrType = HTML
html
  Mimetype = "text/html",
  SvrType = HTML
jsp
  Mimetype = "application/jsp",
  SvrType = JSV

*LOGGING
accesslog
  Format = "default",
  Filename = "log/access_%Y%%M%%D%.log",
  Option="Sync"
errorlog
  Format = "ERROR",
  Filename = "log/error_%Y%%M%%D%.log"
systemlog
  Format = "SYSLOG",
  Filename = "log/system_%Y%%M%%D%.log"
```

A.2.1. JEUS 6 연동 환경설정

다음은 JEUS 6 환경설정 파일 예제로 WebtoB 연결은 <webtob-listener>에 설정한다.

<WEBMain.xml>

```
<!-- JEUS의 설정파일 -->
<?xml version="1.0"?>
<!DOCTYPE web-container PUBLIC "-//Tmax Soft., Inc.//DTD WEB Main Config 4.0//EN"
                       "http://www.tmaxsoft.com/jeus/dtd/4.0/web-main-config.dtd">
<web-container>
    <context-group>
        <group-name>MyGroup</group-name>
        <group-docbase>webapps</group-docbase>
        <session-config>
            <timeout>20</timeout>
            <shared>true</shared>
        </session-config>
        <logging>
            <error-log>
                <target>stdout</target>
                <level>information</level>
                <buffer-size>0</buffer-size>
                <valid-day>1</valid-day>
            </error-log>
            <user-log>
                <target>file</target>
                <buffer-size>0</buffer-size>
                <valid-day>1</valid-day>
            </user-log>
            <access-log>
                <target>file</target>
```

```
                <buffer-size>0</buffer-size>
                <valid-day>1</valid-day>
                <log-format>
                    <time-format>default</time-format>
                </log-format>
            </access-log>
        </logging>
        <context>
            <context-name>examples</context-name>
            <context-path>/examples</context-path>
        </context>
        <context>
            <context-name>test</context-name>
            <context-path>/test</context-path>
        </context>
        <webserver-connection>
<!--
            <http-listener>
                <listener-id>http1</listener-id>
                <port>8989</port>
                <output-buffer-size>8192</output-buffer-size>
                <thread-pool>
                    <min>25</min>
                    <max>30</max>
                    <step>2</step>
                    <max-idle-time>1000</max-idle-time>
                </thread-pool>
            </http-listener>
-->
            <webtob-listener>
                <listener-id>webtob1</listener-id>
                <port>9999</port>
                <hth-count>1</hth-count>
                <webtob-address>192.168.1.43</webtob-address>
                <registration-id>MyGroup</registration-id>
                <thread-pool>
                    <min>10</min>
                    <max>10</max>
                    <step>2</step>
                </thread-pool>
                <disable-pipe>true</disable-pipe>
            </webtob-listener>
        </webserver-connection>
    </context-group>
</web-container>
```

A.2.2. 내장 Servlet Engine(JEUS 8) 연동 환경설정

다음은 내장 Servlet Engine을 연동하는 경우 환경설정 파일의 예제로 WebtoB 연결은 <webtob-connector>에 설정한다.

<domain.xml>

```
<!-- JEUS 8(내장 servlet engine)의 설정파일 -->
<?xml version="1.0" encoding="UTF-8"?>
<domain xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="7.0">
```

```
   <id>8322013</id>
   <admin-server-name>adminServer</admin-server-name>
   <servers>
      <server>
         <name>adminServer</name>

         <listeners>
            <base>base</base>
            <listener>
               <name>base</name>
               <listen-address>0.0.0.0</listen-address>
               <listen-port>9736</listen-port>
               <use-dual-selector>false</use-dual-selector>
               <backlog>128</backlog>
               <read-timeout>30000</read-timeout>
               <reserved-thread-num>0</reserved-thread-num>
            </listener>
            <listener>
               <name>http-server</name>
               <listen-port>8088</listen-port>
               <use-dual-selector>false</use-dual-selector>
               <backlog>128</backlog>
               <read-timeout>30000</read-timeout>
               <reserved-thread-num>0</reserved-thread-num>
            </listener>
         </listeners>

         <web-engine>
            <web-connections>
               <webtob-connector>
                  <name>webtob</name>
                  <output-buffer-size>8192</output-buffer-size>
                  <postdata-read-timeout>30000</postdata-read-timeout>
                  <max-post-size>-1</max-post-size>
                  <max-parameter-count>-1</max-parameter-count>
                  <max-header-count>-1</max-header-count>
                  <max-header-size>-1</max-header-size>
                  <max-querystring-size>8192</max-querystring-size>
                  <network-address>
                     <port>9999</port>
                     <ip-address>localhost</ip-address>
                  </network-address>
                  <thread-pool>
                     <number>10</number>
                     <thread-state-notify>
                        <max-thread-active-time>0</max-thread-active-time>
                        <interrupt-thread>false</interrupt-thread>
                        <active-timeout-notification>false</active-timeout-notification>
                        <notify-threshold-ratio>0.0</notify-threshold-ratio>
                        <restart-threshold-ratio>0.0</restart-threshold-ratio>
                     </thread-state-notify>
                  </thread-pool>
                  <hth-count>1</hth-count>
                  <registration-id>MyGroup</registration-id>
                  <reconnect-interval>5000</reconnect-interval>
               </webtob-connector>
<!--
               <http-listener>
                  <name>http1</name>
```

```
                <output-buffer-size>8192</output-buffer-size>
                <postdata-read-timeout>30000</postdata-read-timeout>
                <max-post-size>-1</max-post-size>
                <max-parameter-count>-1</max-parameter-count>
                <max-header-count>-1</max-header-count>
                <max-header-size>-1</max-header-size>
                <max-querystring-size>8192</max-querystring-size>
                <server-listener-ref>http-server</server-listener-ref>
                <thread-pool>
                   <min>10</min>
                   <max>20</max>
                   <max-idle-time>300000</max-idle-time>
                   <max-queue>-1</max-queue>
                </thread-pool>
                <keep-alive>true</keep-alive>
                <server-access-control>false</server-access-control>
             </http-listener>
-->
            </web-connections>
         </web-engine>

      </server>
   </servers>
</domain>
```

Appendix B: CGI 활용 예제

본 부록에서는 CGI의 동작 과정과 C와 Perl을 사용한 게시판 구현 예제를 설명한다.

B.1. 개요

단순히 보여주기 위한 HTML에서 진일보한 형태인 CGI의 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 보인다.

게시판을 예로 들면 사용자가 글을 남길 수 있는 웹 게시판들은 사용자가 쓴 글을 데이터로 받은 다음 이 데이터를 이용해 다시 HTML 문서를 생성하여 브라우저에게 전달한다. 게시판에 있어서 CGI의 역할은 사용자가 자판으로 치는 문자를 HTML 문서로 변환하는 역할을 한다. 브라우저는 이미 작성된 HTML을 보여주는 역할만 하는 것이므로 미리 작성하여 준 화면만을 보여줄 수 밖에 없는데 CGI를 통해 우리는 이것에 실시간으로 접근하여 글을 남길 수 있다. 게시판 프로그램은 웹 페이지에 직접 쓸 수 있는 "연필"과 같은 기능을 하는 것이다.

본 부록에서는 C를 이용한 게시판과 Perl을 이용한 방명록을 예제로 하여 웹 에서의 CGI를 이용한 서비스를 구현해본다.

B.2. C를 이용한 게시판 구현

UNIX 환경에서 C를 이용한 게시판 작성을 예제로 한다. 간단한 글과 주소를 남길 수 있는 이 웹 페이지는 "board.html"에서 "board.cgi"를 호출해서 구현한다.

B.2.1. 환경 파일 작성

다음은 환경 파일 예제이다.

<sample_cgi.m>

```
#Configuration file  ( sample_cgi.m )

*DOMAIN
webtob

*NODE
webmain         WEBTOBDIR = "$WEBTOBDIR",
                SHMKEY = 72000, HTH=1,
                DOCROOT = "docs/",
                PORT = "8080"

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig          NODENAME = webmain, SvrType = CGI
```

```
*SERVER
cgi           SVGNAME = cgig,  MinProc = 1, MaxProc = 5

*URI
uri1          Uri = "/cgi-bin/", SvrName = cgi, Svrtype = CGI

*ALIAS
alias1        URI = "/cgi-bin/",
              RealPath = "${WEBTOBDIR}/cgi-bin/"
```

B.2.2. 화면 코드 작성

다음은 화면 코드 작성 파일 예제이다.

<board.html>

```
#board.html
<HTML>
<HEAD><TITLE>WebToB Board</TITLE></HEAD>
<BODY>
<H2><big>WebToB 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>
```

B.2.3. CGI 소스코드 작성

다음은 CGI 소스코드 예제이다.

<board.c>

```
#첨부 3: board.c
#include "qDecoder.h"

int strcheck(char *str) {
  /* 문자열이 NULL이거나 길이가 0인지 체크 */
 if (str == NULL)
        return 0;
 if (strlen(str) == 0)
        return 0;
 return 1;
}

int main(void) {
  char *name, *title, *doc;
  char *email, *homepage;

  /* 입력값들을 얻어냄 */
  name = qValue("writer");
  title = qValue("title");
  doc = qValue("doc");
  email = qValue("email");
  homepage = qValue("homepage");

  /* 입력값들이 올바른지 검사 */
  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 !");

  /* 여기서 게시판에 추가합니다. */
  /* 처리 결과 출력 - 입력 확인 */
  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();
}
```

B.2.4. 작업 및 결과 확인

작업 순서는 다음과 같다.

1. 환경 파일인 sample_cgi.m을 시스템에 맞게 수정한다.
2. HTML 문서를 DocRoot로 옮긴다.
3. board.html의 ACTION=”/cgi-bin/board.cgi” 부분이 자신의 환경에서 board.cgi가 있는 곳으로 설정한다.
4. board.c 및 qDecoder.h, qDecoder.c를 CGI가 구동되는 디렉터리로 옮긴다.
5. board.c를 컴파일하면 board.cgi가 생성된다(cc 또는 gcc가 설치되어 있어야 한다).

   ```
   $ cc –o board.cgi board.c qDecoder.c
   ```

   또는

   ```
   $ gcc board.c qDecoder.c –o board.cgi
   ```

6. 생성된 board.cgi의 permission을 755로 설정한다.

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

7. WebtoB를 기동한 후 브라우저를 띄우고 게시판을 호출한다(IP 주소는 현재 WebtoB가 구동되고 있는 머신의 것을 사용한다).

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

8. 실행한 화면은 다음과 같다.

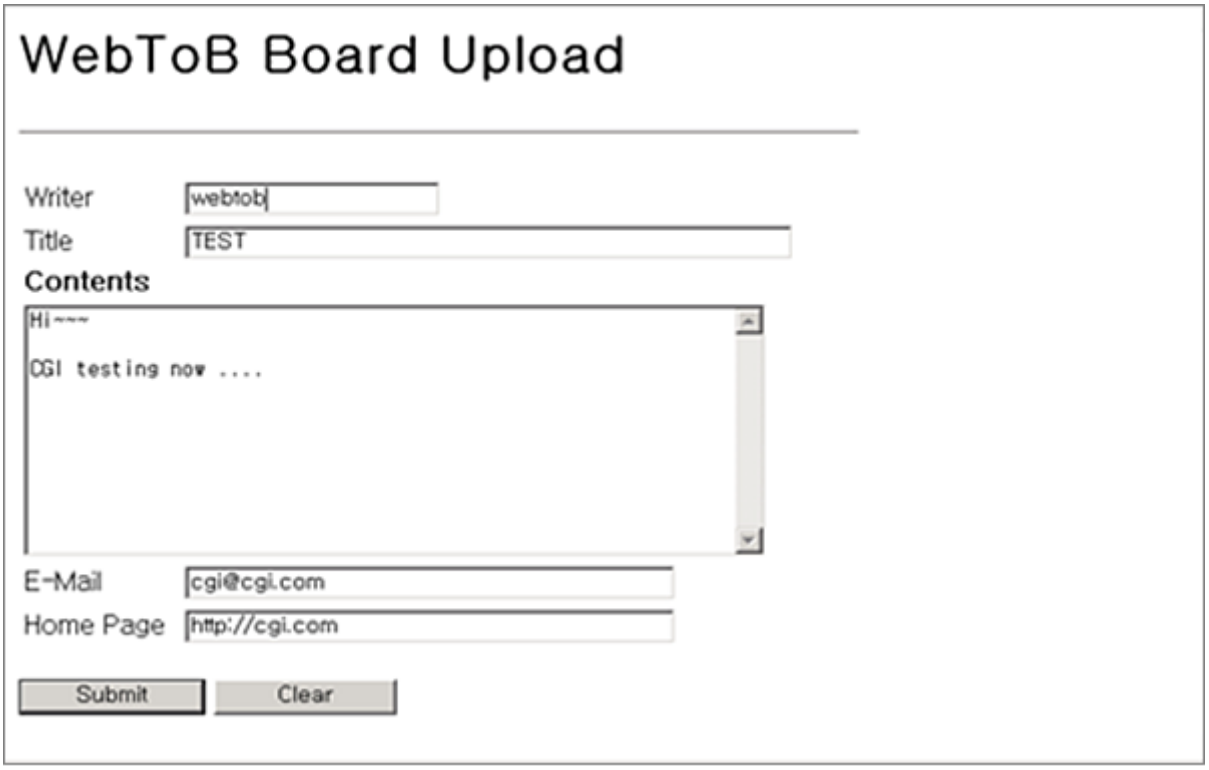


그림 7. C를 이용한 게시판 입력 화면

위와 같이 내용을 입력하고 **[submit]** 버튼을 클릭하면 board.cgi가 호출되어 아래와 같이 게시판에 입력된 화면이 생성된다.

그림 8. C를 이용한 게시판 결과 화면

B.3. Perl을 이용한 게시판 구현

Perl이란 언어는 스크립팅 언어이다. 따라서 컴파일 과정이 필요없이 바로 수행이 가능하기 때문에 프로그램의 작성과 디버깅이 매우 쉽다. 이런 이유로 스크립팅 언어가 선호된다. Perl은 무료로 다운로드가 가능하다. Perl을 사용하기 위한 기초 준비작업은 설명하지 않는다.

본 절에서는 WebtoB에서 Perl을 기동하기 위한 환경 파일과 Perl로 구현한 방명록에 대해 설명한다.

B.3.1. 환경 파일 작성

다음은 환경설정 파일에 대한 예제이다.

<sample_Perl.m>

```
#Configuration file ( sample_perl.m )

*DOMAIN
webtob1

*NODE
webmain          WEBTOBDIR = "/usr/local/webtob",
                 SHMKEY = 72000, HTH=1,
                 DOCROOT = "/usr/local/webtob/docs",
                 PORT = "9989"

*HTH_THREAD
hworker
  WorkerThreads = 16

*SVRGROUP
cgig           NODENAME = webmain,
               SvrType = CGI

*SERVER
cgi            SVGNAME = cgig,
               MinProc = 1,
               MaxProc = 5

*URI
uri1           Uri = "/cgi-bin/",
               SvrName = cgi,
               Svrtype = CGI

*ALIAS
alias1         URI = "/cgi-bin/",
               RealPath = "/usr/local/webtob/docs/cgi-bin/"
```

B.3.2. Perl 스크립트 작성

다음은 Perl 스크립트 작성 파일 예제이다.

<guestbook.cgi>

```
#guestbook.cgi

#!/usr/local/bin/perl

# 기준 URL (보통 홈페이지 주소)
$Base_href      = 'http://webmain.tmax.co.kr:9081/';

# $base_href 를 기준한 이 화일의 path
$Cgi_file       = 'cgi-bin/guestbook.cgi';
```

```
# lock 디렉터리 path
$Lock_dir       = 'lock/guestbook';

# BBS 데이타가 담기는 화일
$Bbs_file       = 'book_data';

# 링크할 제목
$Link_name      = 'Back to Home';
# 링크할 URL ($base_href 기준)
$Link_url       = './';

# 제목
$Bbs_title      = 'Guest Book';

# 관리자 이름
$Bbsmaster_name = 'webtob';

# 관리자 email 주소
$Bbsmaster_email = 'webotb@tmax.co.kr';

# 각 메시지의 최대 크기 (byte) (SPAM 방지용)
$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 ########################################
```

B.3.3. 작업 및 결과 확인

작업 순서는 다음과 같다.

1. 환경 파일인 sample_perl.m을 시스템에 맞게 수정한다.
2. $which perl로 Perl의 경로를 얻어온다.
3. guestbook.cgi를 편집창을 열어 시스템에 맞게 수정한다.

4. guestbook.cgi를 CGI가 구동되는 디렉터리로 옮기고 permission을 755로 설정한다.

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

5. guestbook.cgi의 데이터를 저장하기 위한 book_data 파일을 생성하고 permission을 666으로 설정한다.

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

6. 마찬가지로 lock 디렉터리를 생성하고 permission을 777로 설정한다.

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

7. 시스템을 기동한 후 브라우저에서 다음과 같이 요청한다(IP 주소는 현재 WebtoB가 구동되고 있는 머신의 것을 사용한다).

```
http://IP address:port/cgi-bin/guestbook.cgi
```

8. 실행된 화면은 다음과 같다.

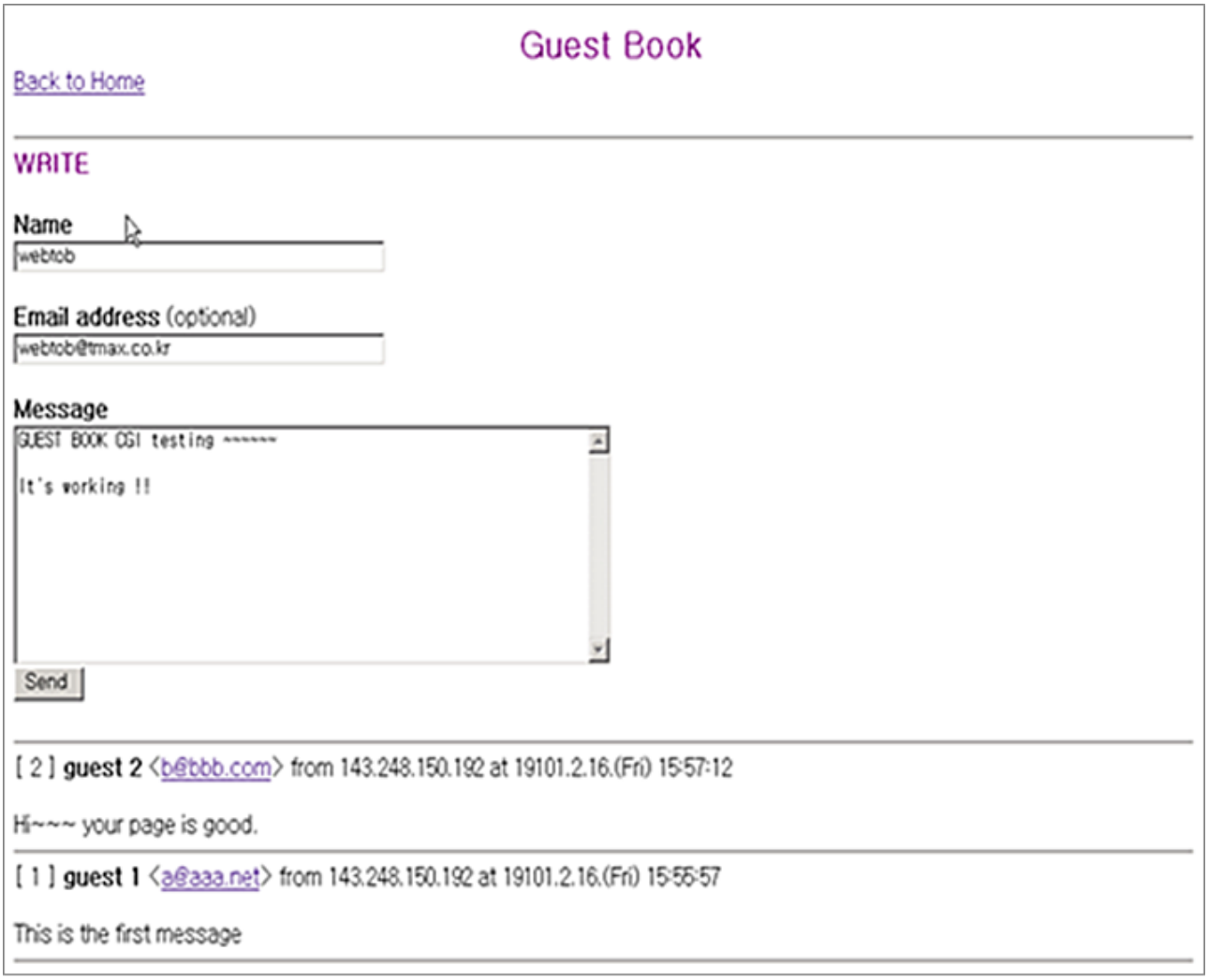


그림 9. Perl을 이용한 게시판 입력 화면

위와 같은 방명록 화면이 뜬다면, 요구하는 정보를 간단히 쓴 다음 **[Send]** 버튼을 클릭해서 방명록에 기록을 남긴다. guest 2가 남긴 말 위에 webtob가 남긴 말이 추가된다. 방문자들이 남긴 글은 book_data 파일에 남는다.

실행된 화면은 다음과 같다.

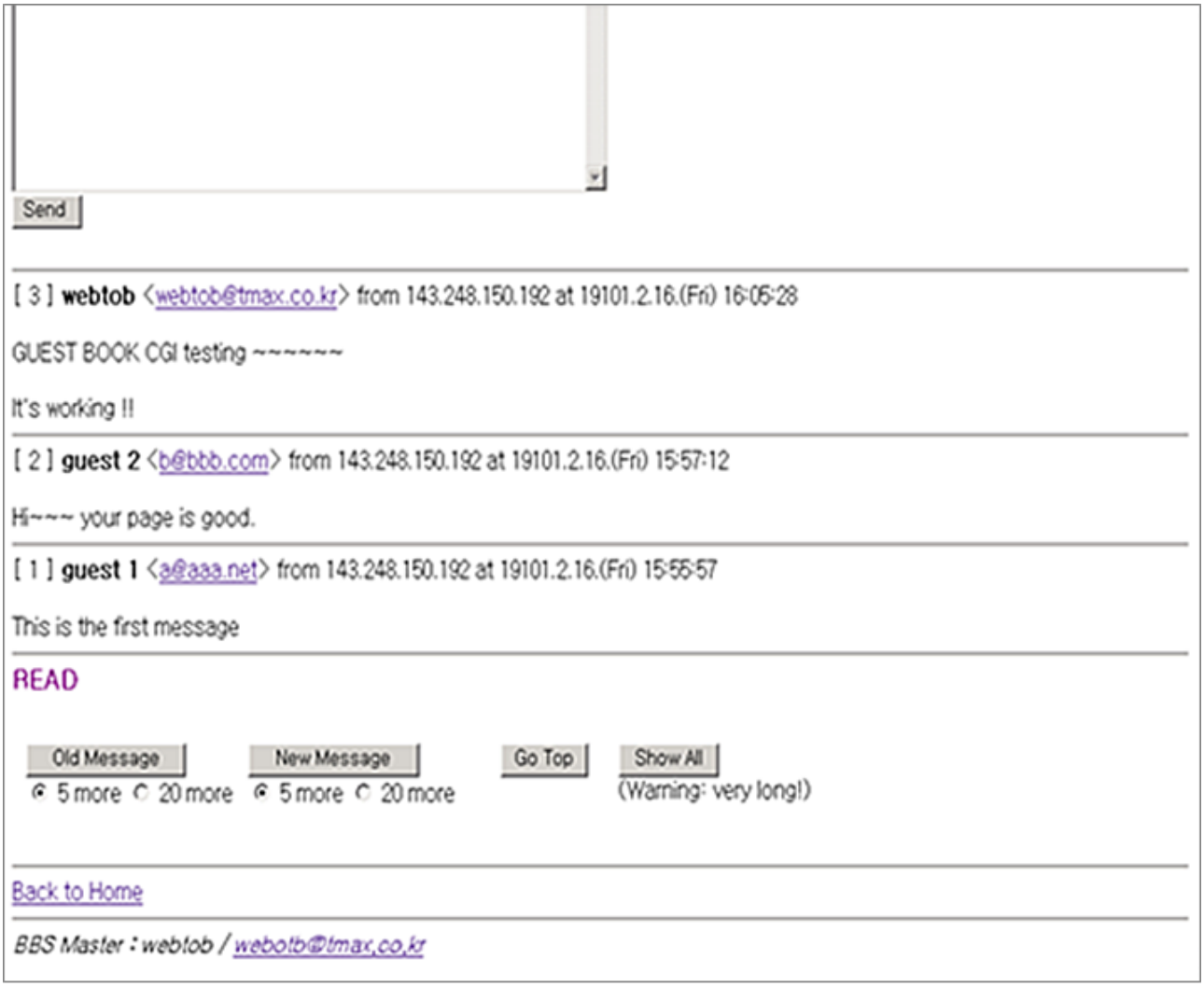


그림 10. Perl을 이용한 게시판 결과 화면

Appendix C: Tmax 연동

본 부록에서는 WebtoB와 Tmax 연동을 위한 클라이언트 프로그램의 작성 방법과 컴파일, 연동 실행 방법을 설명한다.

C.1. Tmax 연동을 위한 클라이언트 프로그램 작성

Tmax 연동을 위한 환경설정에서 제시한 apsl.m에 대한 클라이언트 프로그램을 아래에 제공한다.

본 예제는 Tmax의 sample 서버 프로그램인 svr2의 TOUPPER 서비스를 이용하는 클라이언트 프로그램으로서 wbsession.c와 wbquery.c로 이루어져 있으며 wbquery.c에 tmax.env의 경로를 지정해야 한다.

Tmax는 별도로 설치해야 하며 Tmax 사용과 sample 서버 svr2의 이용에 대해서는 Tmax 매뉴얼을 참고한다.

다음은 wbsession.c 파일이다.

<wbsession.c>

```
#include <stdio.h>
#include <string.h>
#include "../usrinc/wbapi.h"

wbsession(WBSVCINFO *rqst) {
 int len;
 char *value, *value2;
 char *name;
 SESSION *session;

 name = wbGetData( rqst, "name" );
 session= wbGetSession( rqst );
 session = wbSessionSetValue( session, "name", name, strlen(name) );
 value = wbSessionGetValue( session , "name", &len );

 wbSendRedirect( rqst, "/svct/query" );
 wbReturn( rqst, WBSUCCESS );
}
```

다음은 wbquery.c 파일이다.

<wbquery.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/wbapi.h>

int flag=0;

wbSvrInit( int argc, char *argv[] ) {
 fprintf( stdout, "query SERVER Start!\n" );
```

```
 if( test_tpstart() == -1 ) {
     fprintf( stdout,"tpstart failed!!! %s \n",tpstrerror(tperrno) );
 }
 return 0;
}

wbSvrDone() {
 fprintf( stdout, "query SVR DONE!\n" );
 tpend();
}

query(WBSVCINFO *rqst) {
  int len;
  char *name;
  SESSION *session;
  session= wbGetSession( rqst );

  name = wbSessionGetValue( session , "name", &len );
  wbPutHdr(rqst,"Content-type","text/html; charset=euc-kr");
  wbPrint( rqst, "<HTML><BODY>" );

  if( query != NULL ) {
      wbPrint(rqst, "<H1>send Data: %s <br></H1>", name);
  }
  else {
      wbPrint( rqst, "<H1>QUERY is null</H1>" );
  }

  if ( test_tpcall(name) == -1 ) {
      wbPrint( rqst, "<H1>tpcall failed : %s</H1>", tpstrerror(tperrno) );
  }
  else {
      wbPrint( rqst, "<H1>recieved Data : %s</H1>",name );
  }

     wbPrint( rqst, "</body></html>" );
     wbReturn( rqst, WBSUCCESS );
}

int test_tpstart() {
  int  rcode;

  rcode = tmaxreadenv( "/data1/gloria/webtob/ap/tmax.env", "TMAX" );

  if ( rcode == -1 ) {
       printf( "tmax readenv failed %s \n", tpstrerror(tperrno));
       return -1;
  }
  rcode = tpstart( (TPSTART_T *)NULL );

  if ( rcode == -1 ){
       printf( "tpstart failed %s \n",tpstrerror(tperrno) );
       return -1;
  }
  return 1;
}

int test_tpcall(char *name) {
  int  rcode;
```

```
  char   *sndbuf, *rcvbuf;
  long   rcvlen, sndlen;

  if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL){
      printf( "sendbuf alloc failed ! %s \n", tpstrerror(tperrno) );
      tpend();
      return -1;
  }
  if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL){
      printf( "recvbuf alloc failed ! %s \n", tpstrerror(tperrno) );
      tpfree( (char *)sndbuf );
      tpend();
      return -1;
  }
  strcpy(sndbuf, name);

  if(tpcall("TOUPPER", sndbuf, 0, &rcvbuf, &rcvlen, 0)==-1){
      printf("Can't send request to service TOUPPER %s \n", tpstrerror(tperrno));
      tpfree( (char *)sndbuf );
      tpfree( (char *)rcvbuf );
      tpend();
      return -1;
  }
  printf( "[rcvbuf:%s]\n", rcvbuf );
  strcpy( name, rcvbuf );
  tpfree( (char *)sndbuf );
  tpfree( (char *)rcvbuf );
  return 1;
}
```

위 소스코드에서 Tmax 환경을 참조하는 부분(tmaxreadenv)을 각자 환경에 맞는 경로로 수정한다.

다음은 tmax.env에 대한 예제이다.

<tmax.env>

```
[TMAX]
TMAXDIR=/data1/gloria/tmax
TMAX_HOST_ADDR=192.168.1.43
TMAX_HOST_PORT=7979
SDLFILE=/data1/gloria/tmax/sample/sdl/tmax.sdl
FDLFILE=/data1/gloria/tmax/sample/fdl/tmax.fdl
TMAX_CONNECT_TIMEOUT=2 [TMAX]
TMAXDIR=/data1/gloria/tmax
TMAX_HOST_ADDR=192.168.1.43
TMAX_HOST_PORT=7979
SDLFILE=/data1/gloria/tmax/sample/sdl/tmax.sdl
FDLFILE=/data1/gloria/tmax/sample/fdl/tmax.fdl
TMAX_CONNECT_TIMEOUT=2
```

C.2. 클라이언트 프로그램 컴파일

클라이언트 프로그램의 컴파일에 사용하는 Makefile은 다음과 같다.

```
#Makefile.c
TARGET  = $(COMP_TARGET)
APOBJS  = $(TARGET).o

WEBTOB_INCDIR = $(WEBTOBDIR)/usrinc
WEBTOB_BINDIR = $(WEBTOBDIR)/bin
WEBTOB_LIBDIR = $(WEBTOBDIR)/lib

###############
# TMAX_LIBDIR #
###############
#32bit Tmax library
TMAX_LIBDIR = $(TMAXDIR)/lib

#64bit Tmax library
#TMAX_LIBDIR = $(TMAXDIR)/lib64

#SDLFILE = demo.s
#SDLDIR = $(WEBTOBDIR)/sdl

##########
# CFLAGS #
##########

#hp 32bit
CFLAGS = -Ae +DA1.1 +DD32 +DS2.0 -O -I$(WEBTOBDIR)
         -L/data1/gloria/tmax/lib

#hp 64bit
#CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(WEBTOBDIR)
          -L/data1/gloria/tmax/lib

#sun 64bit
#CFLAGS = -xarch=v9 -O -I$(WEBTOBDIR)

#Linux
#CFLAGS = -O -I$(WEBTOBDIR)

#ibm 32bit
#CFLAGS = -q32 -O -I$(WEBTOBDIR) -brtl

#ibm 64bit
#CFLAGS = -q64 -O -I$(WEBTOBDIR) -bnoquiet -brtl

########
# LIBS #
########

#hp, sun, linux
LIBS = -laps -lcli

#ibm
#LIBS = -laps -lcli -lz

OBJS   = $(APOBJS) $(SVCTOBJ) $(SDLOBJ)
SVCTOBJ = $(TARGET)_svctab.o
#SDLOBJ  = ${SDLFILE:.s=_sdl.o}
#SDLC    = ${SDLFILE:.s=_sdl.c}
```

```
.SUFFIXES : .v

.c.o:
 $(CC) $(CFLAGS) -c $<

# Server
$(TARGET): $(APOBJS) $(SVCTOBJ)
 $(CC) $(CFLAGS) -L$(WEBTOB_LIBDIR) -L$(TMAX_LIBDIR) -o $(TARGET) $(OBJS) $(LIBS)
 @rm -f *.o
 @rm -f *_svctab.c

$(SVCTOBJ):
 cp $(WEBTOBDIR)/svct/$(TARGET)_svctab.c .
 $(CC) $(CFLAGS) -I$(WEBTOB_INCDIR) -c $(TARGET)_svctab.c

#
clean:
 -rm -f *.o core
```

위의 Makefile.c는 다음의 컴파일을 이용하여 사용한다.

```
#!/bin/ksh
# program compile
#
#main
        Param=$1
        case "$Param" in
                c)      export COMP_TARGET=$2
                        make -f Makefile.c;;
              api)      export COMP_TARGET=$2
                        make -f Makefile.api;;
               pc)      export COMP_TARGET=$2
                        make -f Makefile.pc all;;
            clean)      make -f Makefile.pc clean;;
                *)      echo "Usage: $0 argument";;
        esac
```

다음은 실행에 대한 예제이다.

```
$ compile c wbsession
$ compile c wbquery
```

C.3. Tmax 연동 실행

위 예제를 이용하여 Tmax와의 연동을 실행해 보기 위한 순서는 다음과 같다.

1. WebtoB 환경 파일을 컴파일한다.

   ```
   $ wscfl –i apsl.m
   ```

2. 서비스 테이블을 생성한다.

```
$ wsgst
```

3. 클라이언트 프로그램을 컴파일한다.

```
$ compile c wbsession
$ compile c wbquery
```

4. Tmax의 서버 프로그램을 컴파일한다.

```
$ compile c svr2
```

5. Tmax를 기동한다.

```
$ tmboot
```

6. WebtoB를 기동한다.

```
$ wsboot
```

실행 결과를 보기 위해 다음의 toupper.html과 같은 HTML 페이지를 이용한다.

```
<html>
<head>
 <title>wbGetQueryString</title>
</head>
<body>
<form method=post action="/svct/wbsession">
<table width=370>
<br>
<tr>
          <td> input send Data</td>
                <td><input type=input name=name></td>
                <td><input type=submit value="submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

그림 11. toupper.html

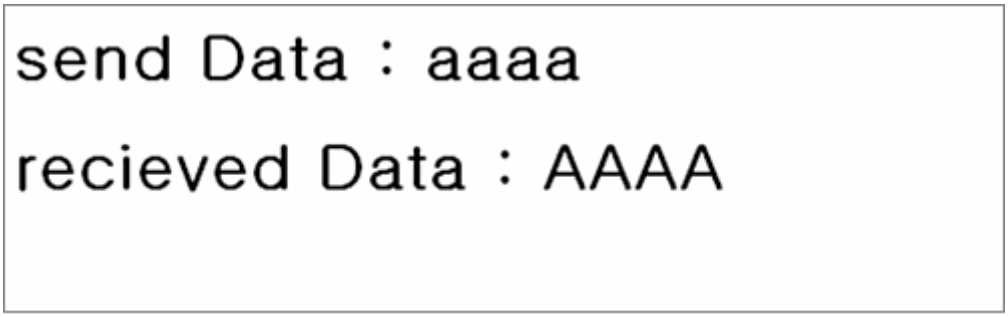


그림 12. TOUPPER 서비스 결과 화면