**TAIN Deploy 사용 설명서**

ver 1.0 (2016.03.17) by 강석. (c) TAIN, Inc.

- 목 차 -

1. 폴더구조

2. bin 폴더의 파일

3. lib 폴더의 파일

4. DEPLOY\_KEY(DEPLOY\_TIME)에 대한 설명

5. DeployServer 데몬 프로그램 실행 및 데몬 프로세스 확인

6. DeployClient 프로그램 실행

7. 배포 확인 방법 by DEPLOY\_KEY(DEPLOY\_TIME)

8. 배포 확인 방법 by diff 명령어(Linux)

9. 소스 보기

10. 로그 보기

11. 테스트 배포 실행 - CI서버에서 만

12. 테스트 배포 실행 - CI서버, App01서버, App02서버 에서

13. 운영 배포 실행 - CI서버, App01서버, App02서버 에서

**1. 폴더구조**

설치 위치는 CI서버, App01서버, App02서버의 아래의 폴더에 설치된다.

/sas/sasuser/sas/deploy

bin : 실행프로그램을 관리하는 폴더

client : Repository에서 자료를 export 하여 MAVEN3을 이용하여 빌드처리를 하는 폴더

여기에 배포를 했던 자료를 보관하여 나중에 복구기능을 할 수 있도록 하였다.

conf : 프로그램 실행이나 처리를 위한 리소스를 보관하는 폴더.

현재는 수정된 pom.xml을 저장하여 관리하고 있다.

lib : 프로그램 실행에 필요한 모든 JAR파일을 저장하는 폴더.

log : 프로그램이 실행하면서 출력되는 메시지를 저장하는 폴더.

만일 배포 처리하는 동안에 에러가 발생하여 정상적으로 배포가 안되었으면

이 폴더에서 해당하는 DEPLOY\_KEY(DEPLOY\_TIME)에 대한 로그를 분석하여

에러 원인을 찾으시길 바랍니다.

server : 배포를 위한 WAR파일을 압축을 풀고 간단한 처리를 하기 위한 서버 측에서

이용하는 폴더.

이 폴더 아래에 app01, app02 폴더는 테스트 배포 시에 사용하는 폴더.

**2. bin 폴더의 파일**

depsvr.sh : 배포작업을 위해서 반드시 실행되어야 하는 프로그램.

이 프로그램은 Background에서 DAEMON 형태로 실행되어야 한다.

이미 실행되고 있으면

DeployServer Process is already running

위와 같은 메시지를 출력하고 실행을 중단한다.

처음 실행하는 경우이면

# DeployServer is starting......

위 메시지를 출력하고 Background에서 DAEMON 형태로 실행된다.

만일 위와 같은 처리 없이 프로세스를 확인하려면

ps -ef | grep DeployServer

명령어를 실행하면 확인할 수 있다.

마지막으로 depsvr.sh은 CI서버, App01서버, App02서버에서 모두 실행되어야

한다.

depcli.sh : 각 서버(CI서버, App01, App02)에서 위 프로그램(depsvr.sh)을 실행하고

이상이 없다면 배포 작업을 실행하는 프로그램이다.

depcli.sh

이 프로그램(depcli.sh)을 실행하면 중간에 멈춤이 없이 아래와 같은 작업들이

자동으로 실행된다.

- CI서버 Repository에서 SASEMARTCMS를 export한다.

- MAVEN3을 이용하여 빌드 처리를 한다.

- WAR파일을 생성하여 관리한다.

- App01서버로 WAR파일을 전송한다.

- App01서버에서 배포 작업하는 동안에 WAS는 중단된다.

- App01서버에 배포 작업을 한다.

- App02서버에서 WAR를 시작한다.

- App02서버로 WAR파일을 전송한다.

- App02서버에서 배포 작업하는 동안에 WAS는 중단된다.

- App02서버에 배포 작업을 한다.

- App02서버에서 WAR를 시작한다.

만일 배포를 했는데 오류가 발생하여 이전 버전을 원복 해야 하는 경우가

발생하면 client폴더에서 관리하는 WAR파일 중에 선택하여 그 파일의

DEPLOY\_KEY(DEPLOY\_TIME)을 아래와 같이 처리하면 된다.

depcli.sh [DEPLOY\_KEY]

여기에서 DEPLOY\_KEY(DEPLOY\_TIME)은 배포 작업을 했을 때의 시간을

YYYYMMDD-HHMMSS 형태로 만든 값이다. 배포 작업을 여러 번 했으면

아래와 같은 WAR파일들이 client 폴더에 여러개 존재한다.

SASEMARTCMS-1.0.0-20160317-115829.war

위 WAR파일의 DEPLOY\_KEY(DEPLOY\_TIME)은 '20160317-115829'이 된다.

그래서 위 WAR파일을 다시 배포작업을 진행하고 싶으면

depcli.sh 20160317-115829

이렇게 실행하면 된다. 이미 이전에 export 한 자료를 갖고 하기 때문에

export 및 빌드 작업은 skip하게 된다.

TR0101.sh : 이 프로그램은 직접 실행하는 프로그램이 아니고 depsvr.sh에서 실행한

DAEMON 프로그램이 실행하는 프로그램이다. 그래서 직접 shell 에서

실행하면 오류가 발생할 수 있다.

CI서버에서 실행하는 프로그램이다.

설명하면 작업을 위하여 client 폴더 아래의 각 폴더들을 지우거나 정리를

하고 Repository 서버로부터 SASEMARTCMS 를 export 한다.

export하는 작업은 약간의 시간이 필요하다.

export를 마치면 MAVEN3을 이용하여 빌드 작업을 한다. 이 빌드 작업이

마치면 SASEMARTCMS/target 폴더가 생기고 그 아래에 WAR 파일이 생성

된다. 그 WAR 파일을 client 폴더로 옮겨 놓는다. 당연히 DEPLOY\_KEY를

붙이고..... 여기까지 작업이 진행된다.

TR1501.sh : 이 프로그램은 직접 실행하는 프로그램이 아니고 depsvr.sh에서 실행한

DAEMON 프로그램이 실행하는 프로그램이다. 그래서 직접 shell 에서

실행하면 오류가 발생할 수 있다.

이 쉘은 App01에서 실행하는 프로그램이다.

먼저 CI서버에서 TR0101.sh 프로그램의 실행결과인 WAR파일을 depcli.sh과

depsvr.sh을 이용하여 파일을 App01서버로 전송되었다. 이것은 내부 명령어로

전송이 되었다. 그 WAR 파일을 풀고 App01서버의 WAS에 배포작업을 한다.

당연히 FILES 폴더는 제외한다.

배포 작업을 하는 동안에 서비스에 장애가 있을 수 있기 때문에 작업 전에는

WAS를 종료하고 배포작업을 마치면 WAS를 시작한다. 이 작업은 30초 내에

마친다.

TR2501.sh : 이 프로그램은 직접 실행하는 프로그램이 아니고 depsvr.sh에서 실행한

DAEMON 프로그램이 실행하는 프로그램이다. 그래서 직접 shell 에서

실행하면 오류가 발생할 수 있다.

이 쉘은 App02에서 실행하는 프로그램이다.

먼저 CI서버에서 TR0101.sh 프로그램의 실행결과인 WAR파일을 depcli.sh과

depsvr.sh을 이용하여 파일을 App02서버로 전송되었다. 이것은 내부 명령어로

전송이 되었다. 그 WAR 파일을 풀고 App02서버의 WAS에 배포작업을 한다.

당연히 FILES 폴더는 제외한다.

배포 작업을 하는 동안에 서비스에 장애가 있을 수 있기 때문에 작업 전에는

WAS를 종료하고 배포작업을 마치면 WAS를 시작한다. 이 작업은 30초 내에

마친다.

**3. lib 폴더의 파일**

ant.jar : ANT 배포를 위한 JAR 파일. 처음에는 사용했다가 MAVEN3로 처리하면

사용하지 않은 파일

commons-logging-1.2.jar : MAVEN3 배포 시에 pom.xml에 포함되어 있지 않아

빌드 에러가 발생하여 ERROR를 발생시킴. 그래서 MAVEN3 빌드 처리하기

전에 포함 했던 JAR 파일

commons-logging-adapters-1.1.jar : MAVEN3 배포 시에 pom.xml에 포함되어 있지 않아

빌드 에러가 발생하여 ERROR를 발생시킴. 그래서 MAVEN3 빌드 처리하기

전에 포함 했던 JAR 파일

commons-net-3.3.0.jar : 배포 작업을 위한 JAR 파일

배포를 위해 실행되는 모든 프로그램은 네트웍 소켓을 사용하여 구성하였다.

그래서 필요한 JAR 파일.

jnlp.jar : 이것은 사용하지 않는다.

log4j-1.2.17.jar : log4j를 위하여 사용하는 JAR 파일

**4. DEPLOY\_KEY(DEPLOY\_TIME)에 대한 설명**

여기서 사용하는 DEPLOY\_KEY 또는 DEPLOY\_TIME이라 하는 값은 YYYYMMDD-HHMMSS 의 형태를 갖고 있다. 이 값은 배포 명령 depcli.sh를 실행한 시간을 나타낸다.

그래서 /sas/sasuser/sas/deploy 폴더 아래에서는 WAR 파일 로그파일 등을 이 값인

DEPLOY\_KEY로 관리하는 것이다.

client 폴더에는

SASEMARTCMS/ <- 작업 폴더

SASEMARTCMS-1.0.0-20160317-115829.war

SASEMARTCMS-1.0.0-20160317-120134.war

SASEMARTCMS-1.0.0-20160317-120317.war

SASEMARTCMS-1.0.0-20160317-121928.war

위와 같은 형태로 WAR 파일을 관리하고

TR0101-20160317-115829.log

TR0101-20160317-120134.log

TR0101-20160317-120317.log

TR0101-20160317-121928.log

TR1501-20160317-115829.log

TR1501-20160317-120134.log

TR1501-20160317-120317.log

TR1501-20160317-121928.log

TR2501-20160317-115829.log

TR2501-20160317-120134.log

TR2501-20160317-120317.log

TR2501-20160317-121928.log

log 폴더는 위와 같은 형태로 파일을 관리한다.

**5. DeployServer 데몬 프로그램 실행 및 데몬 프로세스 확인**

DeployServer 데몬 프로그램은 CI서버, App01서버, App02서버 에서 모두 실행되어야

한다. CI서버에서는 client 폴더에서 작업을 하고 App01서버와 App02서버에서는

server 폴더에서 작업을 한다.

실행되고 있는 DeployServer 데몬 프로세스를 확인하는 방법은

ps -ef | grep DeployServer

위 명령어를 실행하면 관련된 프로세스 정보가 출력된다. 아무런 메시지도 보이지

않으면 DeployServer 데몬 프로그램을 실행하지 않은 것이다.

실행은 당연히 bin 폴더에서

depsvr.sh

이렇게 실행하면 된다.

만일 이미 DeployServer 데몬 프로그램을 실행했다면, 즉 프로세스가 실행되고 있는

중이라면 depsvr.sh 을 실행해도 이상 없다. 이미 실행되고 있다는 메시지를 출력한다.

DeployServer 가 실행되면서 남기는 로그는 bin 폴더에 depsvr.log 파일로 남게 된다.

**6. DeployClient 프로그램 실행**

이 프로그램은 배포 명령어를 각 해당 서버로 전송하여 해당서버에서 실행하도록 하는

프로그램이다.

이 프로그램의 실행은 CI서버에서 실행해야 한다. 실행은 아래와 같다.

depcli.sh

위 명령어는 Repository에 있는 최근 자료를 갖고 배포 작업을 하는 것이다.

당연히 DEPLOY\_KEY도 현재시간으로 되어있을 것이다.

배포가 마치면 client 폴더에는 최근 시간으로 WAR파일이 생성되어 있을 것이다.

여기에 또 다른 폴더 SASEMARTCMS 폴더는 WAR파일을 생성하기 위한 작업을

했던 폴더이다.

SASEMARTCMS-1.0.0-[DEPLOY\_KEY].war

여기서 client 폴더에는 이전에 배포작업을 수행한 많는 WAR 파일들이 있을 것이다.

그래서 이전에 배포작업을 했던 자료를 이용하여 원복 하고자 한다면 아래와 같이

간단히 DEPLOY\_KEY 만 추가하면 된다.

depcli.sh [DEPLOY\_KEY]

그러면 Repository export 과정과 MAVEN3 빌드 과정을 생략하고 WAR파일을 전송하고

배포작업(정리작업) 만을 수행한다. 당연히 원복 작업 시간은 30초 내로 끝난다.

그런데 DEPLOY\_KEY를 잘못 입력하면 배포작업 처리는 진행하지 않는다.

배포작업 진행 시에 중요한 점은 아무나 배포 작업을 실행하지 않도록 약간의 비번을

걸어 놓았다.

##### input the deploy key...

위와 같은 메시지가 출력이 되면 '**deploy123**'이라 입력하면 된다.

**7. 배포 확인 방법 by DEPLOY\_KEY(DEPLOY\_TIME)**

배포의 대상이 되는 서버는 App01서버와 App02서버 만이다.

그래서 배포 되었으면 WAS를 실행하여 브라우저에서 직접 클릭하면서 확인해야 했다.

그런 불편한 불확실한 점을 없애고자 배포된 자료에 DEPLOY\_KEY를 포함시켰다.

css/

FILES/

html/

images/

js/

menu/

META-INF/

WEB-INF/

**20160317-121928 <- DEPLOY\_KEY(DEPLOY\_TIME)**

code404.jsp

code500.jsp

egovDevIndex.jsp

index.jsp

위는 배포가 마친 상태를 보고 있는 것이다. 위 내용의 원천인 WAR 파일은 client 폴더

SASEMARTCMS-1.0.0-**20160317-121928**.war

인 것이다.

**8. 배포 확인 방법 by diff 명령어(Linux)**

Linux 명령어 diff를 이용한 배포 확인 방법은 실제 운영에 배포하고 확인하는

방법이 아니라 배포가 잘되는지 확인하는 방법으로 먼저 테스트 배포를 실행하고

나서 테스트 배포 폴더와 현재 운영 중인 폴더를 비교하여 누락된 파일이 없는지

확인하는 방법이다. 그래서 이 8번 항목을 하기 위해서는 먼저 테스트 배포를 먼저

해야 한다.

먼저 diff 명령어에 대해 알아보자.

diff -urN [소스1폴더] [소스2폴더]

-u 옵션 : diff의 포맷

-r 옵션 : 해당 폴더 하위 폴더 모두 검색

-N 옵션 : 새로 생성된 파일도 diff에 포함

그럼 다음과 같이 테스트 폴더와 운영 폴더를 가정해보자.

테스트 폴더 :

/sas/sasuser/sas/deploy/server/app01/sas\_webapps/sas.emartcms.war

운영 폴더 :

/sas/sasuser/sas/sas\_webapps/sas.emartcms.war

두 폴더를 비교하는 diff 명령어는 다음과 같다.

diff -urN /sas/sasuser/sas/deploy/server/app01/sas\_webapps/sas.emartcms.war

/sas/sasuser/sas/sas\_webapps/sas.emartcms.war | grep "diff -urN"

위와 같이 하면 비교하는 파일의 내용 중에 틀린 부분이 있으면 파일명을 출력한다.

**9. 소스 보기**

주의 깊게 보기 바랍니다. 중요 부분은 빨간색으로 표시하였습니다.

depsvr.sh : CI서버, App01서버, App02서버 에서 실행해야 함

|  |
| --- |
| #!/bin/ksh  #------------------------------------------------------  if [ `ps -ef | grep DeployServer | grep -v grep | grep -v tail | wc -l` -ne 0 ]; then  echo '################################################'  echo '# DeployServer Process is already running. #'  echo '################################################'  exit 0  else  이미 실행이 되었으면 메시지만 출력하고 종료한다.  echo '# DeployServer is starting.......'  fi  #------------------------------------------------------  MAIN\_CLASS=org.apache.commons.net.deploy.server.TainServerMain  #------------------------------------------------------  JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86\_64  JRE\_HOME=$JAVA\_HOME  JAVA\_EXE=${JAVA\_HOME}/bin/java  #------------------------------------------------------  배포를 위한 HOME 폴더.  테스트 배포는 모드 이 폴더 아래에서 처리되고 실행된다.  테스트배포 환경임.  DEPLOY\_HOME=/sas/sasuser/sas/deploy  LIB\_HOME=${DEPLOY\_HOME}/lib  #------------------------------------------------------  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/lib/tools.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/rt.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/resources.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/jsse.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/jce.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/charsets.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/dnsns.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/localedata.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunjce\_provider.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunmscapi.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunpkcs11.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/ant.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-logging-1.2.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-logging-adapters-1.1.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-net-3.3.0.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/jnlp.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/log4j-1.2.17.jar  #------------------------------------------------------  # OPTION="-Xss256K"  OPTION="-Xms512m -Xmx1024m"  OPTION="${OPTION} -Dname=DeployServer"  OPTION="${OPTION} -Ddev.serial=KK28RWYXBC1067AS"  OPTION="${OPTION} -Ddev.author=Kang\_Seok"  OPTION="${OPTION} -Ddev.version=jdk1.7.0\_79"  OPTION="${OPTION} -Dtain.job.seq.range=1-1"  # OPTION="${OPTION} -Dtain.job.seq.range=1-6"  OPTION="${OPTION} -Dtain.date.format=yyyyMMdd-HHmmss"  OPTION="${OPTION} -Dtain.client.host=127.0.0.1"  OPTION="${OPTION} -Dtain.client.port=2025"  OPTION="${OPTION} -Dtain.client.deploy.file.name=${DEPLOY\_HOME}/client/SASEMARTCMS-1.0.0-YYYYMMDDHHMMSS.war"  OPTION="${OPTION} -Dtain.client.exec.cmd=${DEPLOY\_HOME}/bin/TR0101.sh"  OPTION="${OPTION} -Dtain.client.exec.log=${DEPLOY\_HOME}/log/TR0101-YYYYMMDDHHMMSS.log"  OPTION="${OPTION} -Dtain.server.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.port=2025"  OPTION="${OPTION} -Dtain.server.app01.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.app01.port=2025"  OPTION="${OPTION} -Dtain.server.app02.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.app02.port=2025"  OPTION="${OPTION} -Dtain.server.deploy.file.name=${DEPLOY\_HOME}/server/SASEMARTCMS-1.0.0-YYYYMMDDHHMMSS.war"  OPTION="${OPTION} -Dtain.server.app01.exec.cmd=${DEPLOY\_HOME}/bin/TR1501.sh"  OPTION="${OPTION} -Dtain.server.app01.exec.log=${DEPLOY\_HOME}/log/TR1501-YYYYMMDDHHMMSS.log"  OPTION="${OPTION} -Dtain.server.app02.exec.cmd=${DEPLOY\_HOME}/bin/TR2501.sh"  OPTION="${OPTION} -Dtain.server.app02.exec.log=${DEPLOY\_HOME}/log/TR2501-YYYYMMDDHHMMSS.log"  #------------------------------------------------------  nohup ${JAVA\_EXE} -cp ${CLASSPATH} ${OPTION} ${MAIN\_CLASS} >> depsvr.log 2>&1 &  # ${JAVA\_EXE} -cp ${CLASSPATH} ${OPTION} ${MAIN\_CLASS}  #------------------------------------------------------  이부분은 Background에서 실행되는 데몬 프로세스를 설정하는 부분이다. 로그는 depsvr.log에 쌓인다. |

depcli.sh : CI서버 에서 실행해야 함.

|  |
| --- |
| #!/bin/ksh  #------------------------------------------------------  MAIN\_CLASS=org.apache.commons.net.deploy.client.TainClientMain  배포를 위한 HOME 폴더.  테스트 배포는 모드 이 폴더 아래에서 처리되고 실행된다.  테스트배포 환경임.  #------------------------------------------------------  JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86\_64  JRE\_HOME=$JAVA\_HOME  JAVA\_EXE=${JAVA\_HOME}/bin/java  #------------------------------------------------------  DEPLOY\_HOME=/sas/sasuser/sas/deploy  LIB\_HOME=${DEPLOY\_HOME}/lib  #------------------------------------------------------  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/lib/tools.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/rt.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/resources.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/jsse.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/jce.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/charsets.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/dnsns.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/localedata.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunjce\_provider.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunmscapi.jar  CLASSPATH=${CLASSPATH}:${JAVA\_HOME}/jre/lib/ext/sunpkcs11.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/ant.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-logging-1.2.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-logging-adapters-1.1.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/commons-net-3.3.0.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/jnlp.jar  CLASSPATH=${CLASSPATH}:${LIB\_HOME}/log4j-1.2.17.jar  #------------------------------------------------------  이 부분은 최근 소스 배포와 원복을 처리하는 부분이다. 인자로 DEPLOY\_KEY를 주면은 DEPLOY\_KEY에 해당하는 WAR파일이 배포(원복)된다  # OPTION="-Xss256K"  OPTION="-Xms512m -Xmx1024m"  OPTION="${OPTION} -Dname=DeployClient"  OPTION="${OPTION} -Ddev.serial=KK28RWYXBC1067AS"  OPTION="${OPTION} -Ddev.author=Kang\_Seok"  OPTION="${OPTION} -Ddev.version=jdk1.7.0\_79"  if [ "$1" = "" ]; then  OPTION="${OPTION} -Dtain.job.seq.range=1-6"  else  OPTION="${OPTION} -Dtain.job.seq.range=3-6"  fi  OPTION="${OPTION} -Dtain.date.format=yyyyMMdd-HHmmss"  OPTION="${OPTION} -Dtain.client.host=127.0.0.1"  OPTION="${OPTION} -Dtain.client.port=2025"  OPTION="${OPTION} -Dtain.client.deploy.file.name=${DEPLOY\_HOME}/client/SASEMARTCMS-1.0.0-YYYYMMDDHHMMSS.war"  OPTION="${OPTION} -Dtain.client.exec.cmd=${DEPLOY\_HOME}/bin/TR0101.sh"  OPTION="${OPTION} -Dtain.client.exec.log=${DEPLOY\_HOME}/log/TR0101-YYYYMMDDHHMMSS.log"  OPTION="${OPTION} -Dtain.server.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.port=2025"  OPTION="${OPTION} -Dtain.server.app01.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.app01.port=2025"  OPTION="${OPTION} -Dtain.server.app02.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.app02.port=2025"  OPTION="${OPTION} -Dtain.server.deploy.file.name=${DEPLOY\_HOME}/server/SASEMARTCMS-1.0.0-YYYYMMDDHHMMSS.war"  OPTION="${OPTION} -Dtain.server.app01.exec.cmd=${DEPLOY\_HOME}/bin/TR1501.sh"  OPTION="${OPTION} -Dtain.server.app01.exec.log=${DEPLOY\_HOME}/log/TR1501-YYYYMMDDHHMMSS.log"  OPTION="${OPTION} -Dtain.server.app02.exec.cmd=${DEPLOY\_HOME}/bin/TR2501.sh"  OPTION="${OPTION} -Dtain.server.app02.exec.log=${DEPLOY\_HOME}/log/TR2501-YYYYMMDDHHMMSS.log"  #------------------------------------------------------  # nohup ${JAVA\_EXE} -cp ${CLASSPATH} ${OPTION} ${MAIN\_CLASS} $1 > /dev/null 2>&1 &  ${JAVA\_EXE} -cp ${CLASSPATH} ${OPTION} ${MAIN\_CLASS} $1  이부분은 App01서버와 App02서버를 설정하는 부분이다. 대충 봐도 알 것이다.  app01.host= xxx.xxx.xxx.xxx  app02.host= xxx.xxx.xxx.xxx  이러면 해당서버의 deploy 폴더로 자료전송된다.  #------------------------------------------------------ |

데몬 프로세스가 아닌 단순한 실행을 한다.

TR0101.sh : CI서버에서 배포 준비시 자동으로 실행됨.

|  |
| --- |
| #!/bin/sh  #--------------------------------------------------------------------------------------------  # DEPLOY\_TIME  if [ "$1" = "" ]; then  인자로 DEPLOY\_TIME이 없으면 프로그램은 종료한다. 메시지를 출력하고...  export DEPLOY\_TIME=00000000-000001  /bin/echo "USAGE : need to add a argument of DEPLOY\_TIME"  exit  else  export DEPLOY\_TIME=$1  fi  echo "DEPLOY\_TIME = $DEPLOY\_TIME"  #-------------------------------------------------------------------------------------------  # set environment  export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86\_64  export M2\_HOME=/sas/sasuser/sas/.jenkins/apache-maven-3.3.3  배포를 위한 HOME 폴더.  테스트 배포는 모드 이 폴더 아래에서 처리되고 실행된다.  테스트배포 환경임.  export SVN\_HOME=/usr/bin  export SVN\_EDITOR=  export PATH=$JAVA\_HOME/bin:$M2\_HOME/bin:$SVN\_HOME:$PATH  export DEPLOY\_HOME=/sas/sasuser/sas/deploy  export JOB\_HOME=$DEPLOY\_HOME/client  export EXPORT\_PATH=$JOB\_HOME/SASEMARTCMS  export WAR\_FILE=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME.war  #--------------------------------------------------------------------------------------------  # version check  svn --version  /bin/echo "-------------- svn version -------------------"  Repository에서 export 하여 SASEMARTCMS를 구성한다.  java -version  /bin/echo "-------------- java version -------------------"  mvn --version  /bin/echo "-------------- maven version -------------------"  # exit  #--------------------------------------------------------------------------------------------  # svn export  /bin/rm -f $WAR\_FILE  /bin/rm -rf $EXPORT\_PATH    svn export svn://matcmsmine01/repo-tasks/SASEMARTCMS $EXPORT\_PATH --username fic01524 --password Kang123!  /bin/echo "-------------- svn export OK !! -------------------"  #--------------------------------------------------------------------------------------------  # pom.xml  export LIB\_HOME=$EXPORT\_PATH/src/main/webapp/WEB-INF/lib  # mvn install:install-file -DgroupId=kang.tain -DartifactId=ibatis -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/ibatis-2.3.4.726b.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=java-json -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/java-json.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=log4j -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/log4j.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=modules.tain -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/modules.tain.0.151031.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.core -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.core.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.core.nls -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.core.nls.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.rutil -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.rutil.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.security.sspi -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.security.sspi.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.servlet -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.servlet.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.servlet.nls -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.servlet.nls.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.svc.connection -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.svc.connection.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sas.svc.connection.nls -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sas.svc.connection.nls.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=sastpj.rutil -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/sastpj.rutil.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=scsl -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/scsl.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=tdgssconfig -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/tdgssconfig.jar  # mvn install:install-file -DgroupId=kang.tain -DartifactId=terajdbc4 -Dversion=1.0 -Dpackaging=jar -Dfile=$LIB\_HOME/terajdbc4.jar  cp $DEPLOY\_HOME/conf/pom.xml $EXPORT\_PATH  echo "$DEPLOY\_TIME" > $EXPORT\_PATH/src/main/webapp/$DEPLOY\_TIME  /bin/echo "-------------- pom.xml OK !! -------------------"  MAVEN3 빌드처리를 한다.  #--------------------------------------------------------------------------------------------  # clean build  mvn -file $EXPORT\_PATH clean install  /bin/echo "-------------- clean build OK !! -------------------"  #--------------------------------------------------------------------------------------------  # finish  mv $EXPORT\_PATH/target/SASEMARTCMS-1.0.0.war $WAR\_FILE  /bin/echo "-------------- FINISH -------------------" |

TR1501.sh : App01서버에서 배포시 자동으로 실행됨

|  |
| --- |
| #!/bin/sh  #--------------------------------------------------------------------------------------------  # DEPLOY\_TIME  인자로 DEPLOY\_TIME이 없으면 프로그램은 종료한다. 메시지를 출력하고...  if [ "$1" = "" ]; then  export DEPLOY\_TIME=00000000-000001  /bin/echo "USAGE : need to add a argument of DEPLOY\_TIME"  exit  else  export DEPLOY\_TIME=$1  fi  echo "DEPLOY\_TIME = $DEPLOY\_TIME"  #-------------------------------------------------------------------------------------------  # set environment  export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86\_64  export M2\_HOME=/sas/sasuser/sas/.jenkins/apache-maven-3.3.3  export SVN\_HOME=/usr/bin  export SVN\_EDITOR=  export PATH=$JAVA\_HOME/bin:$M2\_HOME/bin:$SVN\_HOME:$PATH  export DEPLOY\_HOME=/sas/sasuser/sas/deploy  export BIN\_PATH=$DEPLOY\_HOME/bin  테스트 배포를 위한 폴더이다. 만일 운영에 배포하려면 WAS\_HOME을 적당하게 수정하면 된다.  export JOB\_HOME=$DEPLOY\_HOME/server  export WAR\_PATH=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME  export WAR\_FILE=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME.war  # TODO 2016.03.16 : change the below in server app01  export WAS\_HOME=$JOB\_HOME/app01/sas\_webapps/sas.emartcms.war  #--------------------------------------------------------------------------------------------  # version check  svn --version  /bin/echo "-------------- svn version -------------------"  java -version  /bin/echo "-------------- java version -------------------"  mvn --version  /bin/echo "-------------- maven version -------------------"  배포가 진행되는 동안에 WAS를 종료하기 위한 부분이다. 테스트 배포가 완전하게 확인이 되면 이 부분을 수정하길 바란다.  # exit  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  # 1. stop WAS  # exit  #--------------------------------------------------------------------------------------------  # 2. decompress  cd $JOB\_HOME  rm -rf $WAR\_PATH  mkdir $WAR\_PATH  cd $WAR\_PATH  jar xvf $WAR\_FILE  # exit  #--------------------------------------------------------------------------------------------  # 3. delete files and folders  이 부분은 FILES 폴더를 제외하고 나머지를 지우는 부분이다.  cd $WAS\_HOME  for ITEM in `ls`  do  if [ "$ITEM" != "FILES" ]; then  rm -rf $ITEM  echo deleted $ITEM  fi  done  #--------------------------------------------------------------------------------------------  # 4. move files and folders  cd $WAS\_HOME  cp -r $WAR\_PATH/\* .  배포가 마치면 WAS를 실행하는 부분이다. 테스트 배포가 완전하게 확인이 되면 이 부분을 수정하길 바란다.  #--------------------------------------------------------------------------------------------  # 5. delete job folders and files  cd $JOB\_HOME  rm -rf $WAR\_PATH  rm -f $WAR\_FILE  #--------------------------------------------------------------------------------------------  # 6. start WAS  #--------------------------------------------------------------------------------------------  # finish  /bin/echo "-------------- FINISH -------------------"  cd $BIN\_PATH |

TR2501.sh : : App02서버에서 배포시 자동으로 실행됨

|  |
| --- |
| #!/bin/sh  #--------------------------------------------------------------------------------------------  # DEPLOY\_TIME  인자로 DEPLOY\_TIME이 없으면 프로그램은 종료한다. 메시지를 출력하고...  테스트 배포를 위한 폴더이다. 만일 운영에 배포하려면 WAS\_HOME을 적당하게 수정하면 된다.  배포가 진행되는 동안에 WAS를 종료하기 위한 부분이다. 테스트 배포가 완전하게 확인이 되면 이 부분을 수정하길 바란다.  if [ "$1" = "" ]; then  export DEPLOY\_TIME=00000000-000001  /bin/echo "USAGE : need to add a argument of DEPLOY\_TIME"  exit  else  export DEPLOY\_TIME=$1  fi  echo "DEPLOY\_TIME = $DEPLOY\_TIME"  #-------------------------------------------------------------------------------------------  # set environment  export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86\_64  export M2\_HOME=/sas/sasuser/sas/.jenkins/apache-maven-3.3.3  export SVN\_HOME=/usr/bin  export SVN\_EDITOR=  export PATH=$JAVA\_HOME/bin:$M2\_HOME/bin:$SVN\_HOME:$PATH  export DEPLOY\_HOME=/sas/sasuser/sas/deploy  export BIN\_PATH=$DEPLOY\_HOME/bin  export JOB\_HOME=$DEPLOY\_HOME/server  export WAR\_PATH=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME  export WAR\_FILE=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME.war  # TODO 2016.03.16 : change the below in server app01  export WAS\_HOME=$JOB\_HOME/app02/sas\_webapps/sas.emartcms.war  #--------------------------------------------------------------------------------------------  # version check  svn --version  /bin/echo "-------------- svn version -------------------"  java -version  /bin/echo "-------------- java version -------------------"  mvn --version  /bin/echo "-------------- maven version -------------------"  # exit  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  #--------------------------------------------------------------------------------------------  # 1. stop WAS  # exit  #--------------------------------------------------------------------------------------------  # 2. decompress  cd $JOB\_HOME  rm -rf $WAR\_PATH  mkdir $WAR\_PATH  cd $WAR\_PATH  jar xvf $WAR\_FILE  # exit  #--------------------------------------------------------------------------------------------  # 3. delete files and folders  이 부분은 FILES 폴더를 제외하고 나머지를 지우는 부분이다.  cd $WAS\_HOME  for ITEM in `ls`  do  if [ "$ITEM" != "FILES" ]; then  rm -rf $ITEM  echo deleted $ITEM  fi  done  #--------------------------------------------------------------------------------------------  # 4. move files and folders  cd $WAS\_HOME  cp -r $WAR\_PATH/\* .  배포가 마치면 WAS를 실행하는 부분이다. 테스트 배포가 완전하게 확인이 되면 이 부분을 수정하길 바란다.  #--------------------------------------------------------------------------------------------  # 5. delete job folders and files  cd $JOB\_HOME  rm -rf $WAR\_PATH  rm -f $WAR\_FILE  #--------------------------------------------------------------------------------------------  # 6. start WAS  #--------------------------------------------------------------------------------------------  # finish  /bin/echo "-------------- FINISH -------------------"  cd $BIN\_PATH |

**10. 로그보기**

로그는 deploy 폴더 아래의 log 폴더에 쌓인다.

테스트 배포시에는 처리를 위한 로그는 CI서버의 deploy/log에 쌓인다.

CI서버 : TR0101-[DEPLOY\_KEY].log

CI서버 : TR1501-[DEPLOY\_KEY].log

CI서버 : TR2501-[DEPLOY\_KEY].log

테스트를 끝내고 운영배포인 경우에는 로그는 각 서버의 deploy/log에 쌓이게 된다.

CI서버 : TR0101-[DEPLOY\_KEY].log

App01서버 : TR1501-[DEPLOY\_KEY].log

App02서버 : TR2501-[DEPLOY\_KEY].log

위 로그는 각 SHELL 스크립트를 실행하는 동안에 발생하는 모든 로그를 기록한 것이다.

**11. 테스트 배포 실행 - CI서버에서 만**

테스트 배포는 CI서버에서 모든 배포과정이 이뤄진다. 그래서 가상의 App01서버와

App02서버 형태를 구성하였다.

deploy

server

app01

sas\_webapps

sas.emartcms.war <- 요기에 배포됨

app02

sas\_webapps

sas.emartcms.war <- 요기에 배포됨

자! 그럼 배포 테스트를 진행해 보자. deploy/bin 폴더로 이동한다.

먼저 데몬 프로그램을 실행한다.

depsvr.sh

데몬 프로그램은 Backgroud로 포트번호 2025번을 사용하는 서버 프로그램이다.

다음은 배포 명령을 생성하는 클라이언트 프로그램을 실행한다.

depcli.sh

위 명령어를 실행하고 'deploy123'을 입력하면 배포가 실행된다. 약 1분 정도 소요된다.

다 마치고 나면 deploy/server 아래의 폴더들을 확인하면 배포되었음을 알 수 있다.

여러 번 배포를 진행해보고 복원도 한번 실행해보기 바란다. 당근 복원은 DEPLOY\_KEY

가 필요한 걸 알 것이다.

**12. 테스트 배포 실행 - CI서버, App01서버, App02서버 에서**

이번 테스트는 동일 서버가 아닌 다른 서버에 배포하는 방법이다. 이 방법을 수행하려면

먼저 CI서버에 있는 depcli.sh을 수정해야 한다. 수정은 아래와 같다.

CI서버 : depcli.sh

|  |
| --- |
| ..........  OPTION="${OPTION} -Dtain.server.host=127.0.0.1"  OPTION="${OPTION} -Dtain.server.port=2025"  OPTION="${OPTION} -Dtain.server.app01.host=xxx.xxx.xxx.xxx"  OPTION="${OPTION} -Dtain.server.app01.port=2025"  OPTION="${OPTION} -Dtain.server.app02.host= xxx.xxx.xxx.xxx "  OPTION="${OPTION} -Dtain.server.app02.port=2025"  OPTION="${OPTION} -Dtain.server.deploy.file.name=${DEPLOY\_HOME}/server/SASEMARTCMS-1.0.0-YYYYMMDDHHMMSS.war"  OPTION="${OPTION} -Dtain.server.app01.exec.cmd=${DEPLOY\_HOME}/bin/TR1501.sh"  OPTION="${OPTION} -Dtain.server.app01.exec.log=${DEPLOY\_HOME}/log/TR1501-YYYYMMDDHHMMSS.log"  .......... |

수정이 다 되었으면 CI서버에서 depcli.sh을 실행한다. 그러면 배포는 App01서버,

App02서버 순으로 배포가 이루어 진다.

App01 : deploy

server

app01

sas\_webapps

sas.emartcms.war <- 요기에 배포됨

App02 : deploy

server

app02

sas\_webapps

sas.emartcms.war <- 요기에 배포됨

이번에도 배포가 정상적으로 이뤄지면 DEPLOY\_KEY를 이용하여 복원을 실행해 본다.

**13. 운영 배포 실행 - CI서버, App01서버, App02서버 에서**

13번 테스트 배포에서 각 서버의 TR1501.sh, TR2501.sh을 수정한다.

정확히 적당히 수정하기 바랍니다.

App01서버 : TR1501.sh

|  |
| --- |
| ..........  export WAR\_FILE=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME.war  # TODO 2016.03.16 : change the below in server app01  export WAS\_HOME=$JOB\_HOME/app02/sas\_webapps/sas.emartcms.war xxx.xxx.xxx.xxx  ..........  # 1. stop WAS  xxx.xxx.xxx.xxx  ..........  # 6. start WAS  xxx.xxx.xxx.xxx  .......... |

App02서버 : TR2501.sh

|  |
| --- |
| ..........  export WAR\_FILE=$JOB\_HOME/SASEMARTCMS-1.0.0-$DEPLOY\_TIME.war  # TODO 2016.03.16 : change the below in server app01  export WAS\_HOME=$JOB\_HOME/app02/sas\_webapps/sas.emartcms.war xxx.xxx.xxx.xxx  ..........  # 1. stop WAS  xxx.xxx.xxx.xxx  ..........  # 6. start WAS  xxx.xxx.xxx.xxx  .......... |

이번 배포는 운영 배포이기 때문에 서비스 하지 않는 시간에 진행하기 바란다.

마찬가지로 운영배포도 실행은 depcli.sh를 실행하면 된다. 그리고 DEPLOY\_KEY가 필요한

복원도 해보면 된다.

배포 처리 끝나면 WAS 폴더 안에 DEPLOY\_KEY 파일이 생성이 되어 있는 걸 확인할

수 있다.

(수고하셨습니다.)

※ 다음 버전은 위 엔진을 이용한 WEB으로 구현을 할 계획입니다. 많은 성원 부탁합니다.