

Applications & Deployment 안내서

JEUS 9

TMAXSOFT

저작권 공지

Copyright 2024. TmaxSoft Co., Ltd. All Rights Reserved.

제한된 권리

이 소프트웨어(JEUS®) 사용설명서와 프로그램은 저작권법과 국제 조약에 의해 보호됩니다. 사용설명서와 프로그램은 TmaxSoft Co., Ltd.와의 사용권 계약 하에서만 사용할 수 있으며, 사용설명서는 사용권 계약의 범위 내에서만 배포 또는 복제할 수 있습니다. 이 사용설명서의 전부 또는 일부를 TmaxSoft의 사전 서면 동의 없이 전자, 기계, 녹음 등의 수단으로 전송, 복제, 배포하거나 2차적 저작물을 작성할 수 없습니다.

이 소프트웨어 사용설명서와 프로그램의 사용권 계약은 어떠한 경우에도 사용설명서 및 프로그램과 관련된 지적 재산권(등록 여부를 불문)을 양도하는 것으로 해석되지 않으며, 브랜드나 로고, 상표 등을 사용할 권한을 부여하지 않습니다. 사용설명서는 오로지 정보 제공만을 목적으로 하며, 이로 인한 계약상의 직접적 또는 간접적 책임을 지지 않습니다. 또한 사용설명서 상의 내용이 법적 또는 상업적인 특정 조건을 만족시킬 것을 보장하지 않습니다. 사용설명서는 제품의 업그레이드나 수정에 따라 예고 없이 변경될 수 있으며, 내용상의 오류가 없음을 보장하지 않습니다.

상표 공지

JEUS®는 TmaxSoft Co., Ltd.의 등록 상표입니다.

Java, Solaris는 Oracle Corporation 및 그 자회사, 관계회사의 등록 상표입니다.

Microsoft, Windows, Windows NT는 Microsoft Corporation의 등록 상표 또는 상표입니다.

HP-UX는 Hewlett Packard Enterprise Company의 등록 상표입니다.

AIX는 International Business Machines Corporation의 등록 상표입니다.

UNIX는 X/Open Company, Ltd.의 등록 상표입니다.

Linux는 Linus Torvalds의 등록 상표입니다.

Noto는 Google Inc.의 상표입니다. Noto 글꼴은 오픈 소스입니다. 모든 Noto 글꼴은 SIL Open Font License, 버전 1.1에 따라 게시됩니다. (<https://www.google.com/get/noto/>)

기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상호, 상표, 또는 등록 상표입니다.

본 사용설명서에 기재된 회사, 시스템, 제품 이름 등에 반드시 상표 표시(™, ®)를 하지는 않습니다.

오픈 소스 소프트웨어 공지

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다.: APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

관련 상세 정보는 제품의 다음 디렉터리에 기재된 사항을 참고하시기 바랍니다. :

\${INSTALL_PATH}/license/oss_licenses

기술 지원

홈페이지: <https://www.tmaxsoft.com>

기술 서비스 센터: 1544-8629

안내서 이력

제품 버전	안내서 버전	발행일	비고
JEUS 9	3.1.1	2024-09-27	-

목차

용어 해설	1
1. 도메인 환경에서 애플리케이션 관리	2
1.1. 애플리케이션 관리	2
1.1.1. 2 Phase Deployment	3
1.1.2. 애플리케이션 ID	5
1.1.3. 애플리케이션 상태	5
1.1.4. 애플리케이션 관리 디렉터리 구조	8
1.1.5. 애플리케이션 대상	10
1.2. Managed Server(MS)에서의 Deploy	12
1.2.1. Run-time Deploy	12
1.2.2. Boot-time Deploy	13
1.2.3. 애플리케이션 동기화	15
1.3. Boot-time Auto Deployment	15
1.4. Auto Redeploy	16
1.5. 디렉터리 모드의 Deploy	17
1.6. 애플리케이션 저장소	17
1.6.1. 애플리케이션 저장소 추가/삭제/조회	17
1.6.2. 애플리케이션 저장소에 있는 애플리케이션 Deploy	19
1.7. path를 지정하여 Deploy	19
1.8. Staging Mode Deploy	20
1.9. 애플리케이션 Undeploy	20
2. Graceful Undeployment와 Redeployment	21
2.1. Graceful Undeployment	21
2.2. Graceful Redeployment	23
2.2.1. Graceful Redeploy를 사용하기 위한 전제 조건	23
2.2.2. Graceful Redeploy에 대한 필수 고려 사항	24
2.2.3. Graceful Redeploy 사용 방법	24
2.2.4. 웹 애플리케이션 Graceful Redeploy	27
2.2.5. EJB 애플리케이션 Graceful Redeploy	28
2.2.6. EAR 애플리케이션 Graceful Redeploy	28
3. 애플리케이션	29
3.1. 모듈과 애플리케이션	29
3.1.1. 모듈	30
3.1.2. 애플리케이션	30
3.2. Deployment Descriptor(DD)	31
3.3. 애플리케이션에서 라이브러리 사용	33
3.3.1. lib/application 디렉터리	33
3.3.2. 공유 라이브러리	35
3.3.3. Library Deployment	39

3.3.3.1. 라이브러리 설치 및 삭제	40
3.3.3.2. 애플리케이션에서 라이브러리 참조	42
4. 애플리케이션 작성 및 Deploy	43
4.1. 애플리케이션 작성	43
4.2. Deploy 명령어	44
4.3. 애플리케이션 제어 및 모니터링	45
4.3.1. 애플리케이션을 도메인에 Install	45
4.3.2. 도메인에서 애플리케이션 Uninstall	46
4.3.3. 애플리케이션 Deploy	46
4.3.3.1. 도메인에 install한 애플리케이션 Deploy	47
4.3.3.2. 애플리케이션 저장소에 있는 애플리케이션 Deploy	50
4.3.3.3. path를 지정하여 Deploy	51
4.3.4. 애플리케이션 Redeploy	51
4.3.5. 애플리케이션 Undeploy	52
4.3.6. 애플리케이션 시작	52
4.3.7. 애플리케이션 중지	53
4.3.8. 서비스 중인 애플리케이션에 서버 추가	54
4.3.9. 서비스 중인 애플리케이션에서 서비스 중인 서버 삭제	55
4.3.10. 애플리케이션 정보 확인	56
4.3.10.1. 콘솔 툴 사용	56
4.4. Staging Mode Deploy	59
4.5. Deployment Plan을 사용한 Deployment	60
4.5.1. Deployment Plan 설정 및 동작 방식	60
4.5.2. Deployment Plan Install	65
4.5.3. Install한 Deployment Plan 확인	66
4.5.4. Deployment Plan을 적용한 Deploy	66
4.5.5. 애플리케이션에 적용된 Deployment Plan 확인	67
4.5.6. Deployment Plan Uninstall	67
4.5.7. Deployment Plan Redeploy	68

용어 해설

Console

GUI 인터페이스와 반대되는 개념의 COMMAND 인터페이스이다(Terminal Window).

DD

Deployment Descriptor의 약어이다.

jeus-application-dd.xml

JEUS Application DD 파일이다.

domain.xml

JEUS WAS 환경 파일이다.

1. 도메인 환경에서 애플리케이션 관리

본 장에서는 도메인 환경에서 애플리케이션이 어떻게 관리되고, Deploy 작업은 어떻게 진행되는지 설명한다. 또한 Managed Server(이하 MS)에서 Deploy 작업 진행에 대해서 설명한다.

1.1. 애플리케이션 관리

Master Server는 도메인을 관리하는 서버이다. 애플리케이션은 도메인 단위로 관리되어야 하기 때문에 Master Server에서는 도메인에 존재하는 모든 애플리케이션을 관리한다.

애플리케이션에 관련된 deploy 명령과 조회 명령은 모두 Master Server를 통해서만 가능하다. Master Server를 기동할 때 애플리케이션을 관리하는 서비스가 시작되고 이 서비스를 통해 서버나 클러스터에 deploy할 수 있다. deploy뿐만 아니라 애플리케이션과 관련된 모든 명령이 Master Server를 통해서만 동작 가능하다. Master Server에 장애가 발생해서 MS가 INDEPENDENT 상태가 된 경우에는 애플리케이션에 deploy하는 명령은 사용할 수 없다. 단, 콘솔 툴(jeusadmin)을 통해 해당 MS에 연결하여 애플리케이션 정보를 확인하는 조회 명령은 사용 가능하다.



MS로 직접 연결하여 애플리케이션 정보를 조회하는 것은 MS가 INDEPENDENT 상태일 때만 가능하다. MS가 INDEPENDENT 상태인 것은 Master Server로 접속할 수 없거나 Master Server에서 MS를 관리할 수 없는 상황이기 때문에 MS로 직접 연결하여 애플리케이션의 정보를 확인한다.

애플리케이션을 서비스할 수 있도록 deploy하기 위해서는 먼저 애플리케이션을 도메인에 install시켜야 한다. 애플리케이션을 install하는 것은 Master Server에 애플리케이션 파일을 설치하는 작업이고, deploy하는 것은 애플리케이션을 서비스할 수 있도록 만드는 작업이다.

• install

애플리케이션 파일을 Master Server로 업로드하는 동작을 의미한다. Master Server에 install된 애플리케이션만이 deploy가 가능하다.

애플리케이션이 install되었다는 것은 다음과 같은 의미를 갖는다.

- JEUS에서 제공하는 방법을 사용하여 애플리케이션 파일을 도메인의 APPLICATION_INSTALL_HOME에 위치시키는 것을 의미한다. APPLICATION_INSTALL_HOME은 도메인에 install된 애플리케이션을 관리하는 저장소이다. 이 저장소는 변경될 수 없고 수동으로 접근해서도 안 된다. install된 애플리케이션을 삭제하거나 새로운 애플리케이션을 수동으로 추가해도 동적으로 반영되지 않는다. Master Server를 재부팅한 경우에는 반영이 되겠지만 이는 절대 권장하는 방법이 아니다.
- 사용자가 지정한 디렉터리에 애플리케이션을 수동으로 위치시키는 것을 의미한다. 사용자가 지정한 디렉터리는 애플리케이션이 저장되는 위치(특정 디렉터리)를 의미한다. 사용자는 이 디렉터리를 애플리케이션 저장소(repository)로 추가하고 여기에 애플리케이션 파일을 수동으로 위치시킬 수 있다. 콘솔 툴의 **add-application-repository**, **remove-application-repository** 명령어를 통해 도메인에 애플리케이션 저장소를 추가, 삭제하는 것이 가능하다.

• deploy

애플리케이션이 서비스될 수 있도록 서버에 애플리케이션을 배포하고 서비스 준비 작업을 수행한다.

사용자가 애플리케이션을 deploy하겠다는 명령을 Master Server로 보내면, Master Server에서는 대상(target)이 되는 서버나 클러스터로 Deploy 작업을 진행한다. 이때 Master Server와 서버에서는 2 Phase Deploy를 진행한다. 2 Phase Deploy에 대한 자세한 설명은 [2 Phase Deployment](#)를 참고한다.

1.1.1. 2 Phase Deployment

애플리케이션 파일을 각 서버로 배포하고 애플리케이션 서비스를 위한 사전 작업을 진행한다. 이 작업이 성공한 후에 애플리케이션을 서비스할 수 있는 상태로 만든다. 이 작업과정을 Master Server에서는 2단계로 나누어서 처리한다. 모든 deploy 명령은 Master Server를 통해서 Master Server에서는 대상이 되는 서버로 다시 명령을 내린다. 사용자가 Deploy 요청을 할 때 대상으로 설정한 모든 서버에 대한 Deploy 성공을 보장할 수 있어야 사용자가 내린 deploy 명령이 성공했다고 볼 수 있다.

따라서 애플리케이션을 **Distribute, Start** 두 단계로 나누어서 진행하고 1단계에서 실패하면 deploy 명령은 실패한 것이고 모든 서버에서 undeploy된다.

- 1단계가 성공하게 되면 애플리케이션에 대한 Distribute 작업과 검증 작업이 성공한 것이기 때문에 잠재적으로 Deploy가 성공한 상태라고 볼 수 있다. 하지만 애플리케이션이 서비스될 수 있는 상태는 아니다.
- 2단계가 성공해야 애플리케이션이 서비스될 수 있는 상태가 된다. 2단계에서는 실패한 서버가 있더라도 재시도해서 애플리케이션이 해당 서버에서 서비스될 수 있는 상태가 될 수 있도록 보장해준다.

이러한 두 단계를 하나의 deploy 명령이 아닌 각각 distribute 명령, start 명령으로 나누어서 수행할 수도 있다.

1단계 (Distribute 단계)

대상이 되는 서버나 클러스터로 애플리케이션을 distribute하고 검증한다. 이 단계를 JEUS에서는 애플리케이션 **Distribute 단계**라고 한다. 이 단계에서는 애플리케이션 파일을 각 서버에 배포하고, 애플리케이션이 서비스될 수 있도록 사전준비 작업과 검증 작업을 수행한다.

실제 Distribute 단계의 작업이 모두 성공하면 서비스를 하기 위한 모든 작업은 끝난 상태이다. EJB 모듈의 경우는 서비스 포트가 열리고 웹 모듈의 경우는 웹 리스너와 웹 모듈의 컨텍스트가 연결된다. 하지만 애플리케이션의 상태는 서비스될 수 없는 DISTRIBUTED 상태이다.

애플리케이션이 DISTRIBUTED 상태일 때 애플리케이션에 서비스를 요청하면 웹 모듈의 경우는 "503 Service Unavailable"이 발생하고, EJB 모듈의 경우는 "Bean이 존재하지 않는다"는 Exception이 발생한다.

EJB 모듈이 DISTRIBUTED 상태일 때 요청이 들어온 경우 발생하는 클라이언트 Exception

```
[2016.08.10 15:28:35][1] [t-1] [JNDI.Context-0073] exception occurred during JNDI operation
<<__Exception__>>
javax.naming.NamingException: jakarta.ejb.EJBException: java.rmi.RemoteException: EJB object is not
read at
jeus.ejb.client.BusinessObjectFactory.getObjectInstance(BusinessObjectFactory.java:89)
  at javax.naming.spi.NamingManager.getObjectInstance(NamingManager.java:304)
  at jeus.jndi.JNSContext.lookupInternal(JNSContext.java:594)
  at jeus.jndi.JNSContext.lookup(JNSContext.java:549)
  at jeus.jndi.JNSContext.lookup(JNSContext.java:538)
  at jeus.jndi.JEUSFailoverContext.lookup(JEUSFailoverContext.java:314)
  at javax.naming.InitialContext.lookup(InitialContext.java:392)
```



```

at test.HelloTest.testHelloBean(HelloTest.java:29)
.....
Caused by: java.rmi.RemoteException: EJB object is not ready at
jeus.ejb.container.RemoteInvocationManagerImpl.beforeInvoke(RemoteInvocationManagerImpl.java:72)
at jeus.ejb.baseimpl.RemoteInvokerServer.preInvoke(RemoteInvokerServer.java:118)
at jeus.ejb.baseimpl.RemoteInvokerServer.invoke(RemoteInvokerServer.java:97)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:305)
at sun.rmi.transport.Transport$1.run(Transport.java:159)
at java.security.AccessController.doPrivileged(Native Method)
at sun.rmi.transport.Transport.serviceCall(Transport.java:155)
at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:535)
at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(TCPTransport.java:790)
at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(TCPTransport.java:649)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:662)

```

HTTP Status : 503 - 503 Service Unavailable

Description	Worker (http1-2): An error occurred while processing the request.

웹 모듈이 DISTRIBUTED 상태일 때 요청을 한 경우 발생하는 에러

사용자가 Deploy 대상으로 설정한 모든 서버와 클러스터로 Distribute 작업이 성공해야 전체 성공이 될 수 있다. 하나라도 실패한 서버가 있다면 사용자가 내린 **deploy** 명령은 실패하고 성공한 서버에는 **undeploy** 명령이 보내지면서 이 단계에서 했던 작업을 모두 rollback한다. Deploy 대상으로 설정한 서버 중 일부 서버가 fail 또는 shutdown된 상태라면 이 또한 Deploy 실패이다. 이 경우 사용자가 Deploy 대상에서 해당 서버를 제외하고 deploy 명령어를 다시 시도해야 한다.

Deploy 대상이 클러스터라면 클러스터 멤버 중 일부 서버가 fail 또는 shutdown된 상태이더라도 Distribute는 성공할 수 있다. 이런 서버들을 제외한 다른 모든 서버에서 Distribute 작업이 성공한다면 이 애플리케이션은 Distribute 성공이다.

2단계 (Start 단계)

대상이 되는 서버나 클러스터에는 이미 애플리케이션이 distribute된 상태일 때 애플리케이션이 서비스될 수 있도록 start시켜준다. 이 단계에서는 애플리케이션을 서비스될 수 있는 상태로 만들어주는 간단한 작업을 수행한다. 이 단계를 JEUS에서는 애플리케이션 **Start 단계**라고 한다.

Start 단계에서는 Distribute 단계에서와는 다르게 대상이 되는 서버나 클러스터 중에서 하나의 서버에서라도 Start 작업이 성공하면 전체 성공으로 간주한다. Distribute 작업을 마치면서 이미 애플리케이션의 검증 작업이 완료되었고, 이미 애플리케이션을 서비스할 수 있는 모든 준비가 갖춰진 것이기 때문이다.

Start 작업이 실패했다는 것은 서버가 잠시 장애상태라고 추정할 수 있다. 이는 서버가 정상화되면 Start 작업이 성공해서 애플리케이션을 서비스할 수 있다는 것을 의미이기도 하다. Start 작업에 실패한 서버는 Master Server에서 별도의 Thread로 5초 간격으로 10번 재시도한다.

재시도도 실패하면 서버가 복구되지 않은 것이므로 수동으로 서버를 복구해야 한다. 서버가 복구된 후에 애플리케이션에 대해 start 명령어를 다시 수행하면 애플리케이션 Start 작업을 실패했던 서버에서 start 명령이 정상적으로 수행된다.

1.1.2. 애플리케이션 ID

JEUS 7부터는 도메인에서 애플리케이션을 관리하기 위해 애플리케이션을 식별할 수 있는 ID를 애플리케이션마다 발급하여 사용하고 있다. 애플리케이션 ID는 도메인에서 애플리케이션을 관리하기 위해 필요한 이름이다. 또한 애플리케이션을 구별할 수 있는 식별자이기 때문에 도메인에서 유일해야 한다.

애플리케이션을 install할 때 ID를 설정할 수 있다. install 명령어를 수행할 때 애플리케이션에 ID를 설정하지 않은 경우에는 애플리케이션 파일 이름으로 ID를 만들어서 사용한다. 예를 들어 examples.ear을 install할 때 ID를 설정하지 않았다면 애플리케이션 ID는 'examples_ear'이 된다. ID로는 알파벳이나 숫자를 사용할 것을 권장한다.

애플리케이션 ID는 deploy/undeploy 등 애플리케이션 제어 명령이나 조회 명령을 할 때 설정해야 한다. 도메인에 별도의 저장소(repository)를 설정하고 해당 위치에 애플리케이션 파일을 위치시킨 경우는 애플리케이션 파일 이름을 그대로 ID로 사용한다.

Master Server에서 애플리케이션 파일을 전송받으면 도메인 디렉터리 하위에 존재하는 애플리케이션 INSTALL_HOME에 ID로 디렉터를 생성하고 그 하위에 애플리케이션 파일을 위치시킨다.

애플리케이션을 install하면 다음의 디렉터리가 생성된다. INSTALL_HOME 디렉터리에 대한 자세한 설명은 [도메인 INSTALL_HOME 디렉터리](#)를 참고한다.

```
INSTALL_HOME/<APPLICATION_ID>/<APPLICATION_FILE>
```

서버에서도 애플리케이션 파일을 전송받을 때나 압축을 풀 때 DEPLOYED 디렉터리 하위에 APPLICATION_ID를 최상위 디렉터리로 두고 그 하위에 파일을 위치시키고 압축을 푼다.



application name은 Java EE 6부터 표준 DD에 정의할 수 있으며, 애플리케이션 서비스를 위해서 반드시 필요한 이름이다. 이 값을 애플리케이션 ID와 혼동하면 안된다.

application name은 애플리케이션이 서비스되고 있는 서버에서만 유일하면 되지만 애플리케이션 ID는 도메인에서 유일해야 한다. application name을 설정하지 않은 경우 확장자를 제외한 애플리케이션 파일 이름이 application name이 된다. application name 설정 방법에 대한 자세한 내용은 "Jakarta EE 9 Platform Specification"을 참고한다.

1.1.3. 애플리케이션 상태

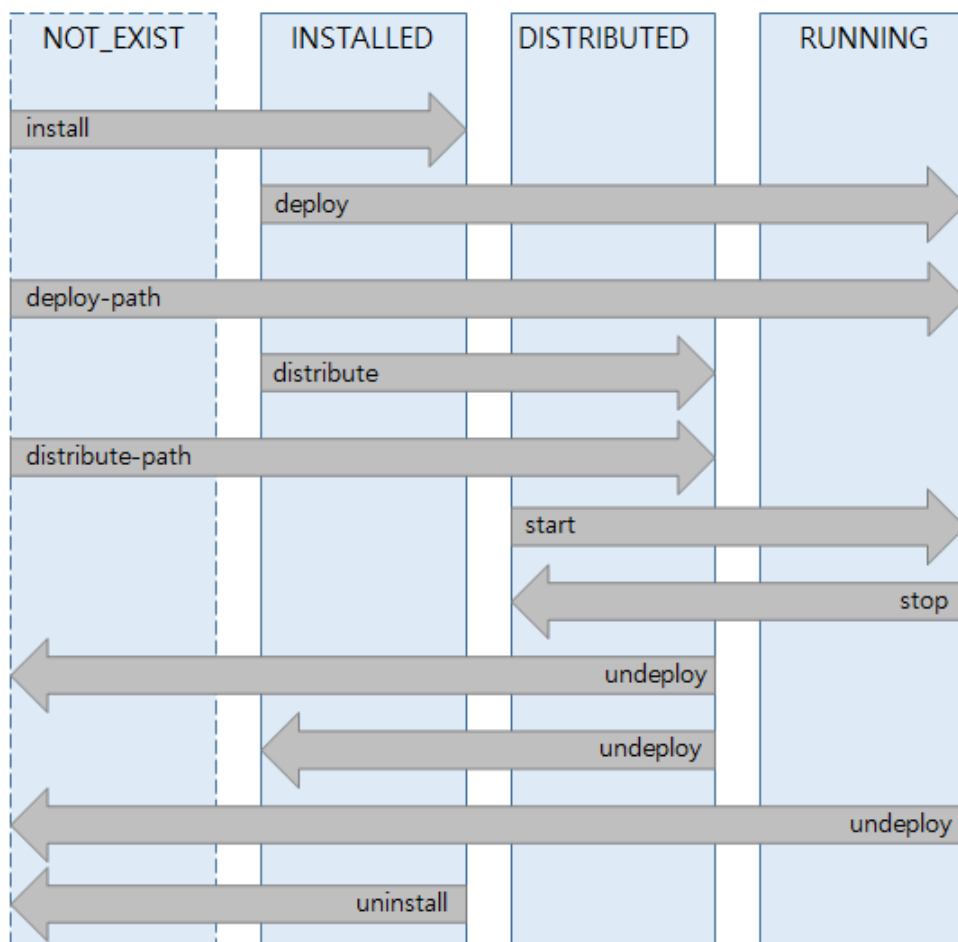
도메인에서는 애플리케이션에 대한 명령을 주관하기 때문에 도메인에서 보여지는 애플리케이션 상태는 target 서버 전체를 총괄하는 상태를 나타낸다. 서버에서는 실제 애플리케이션을 deploy하고 서비스하는 주체이기 때문에

애플리케이션이 어떤 상태인지를 나타낸다.

도메인에서의 애플리케이션 상태

다음은 도메인에서의 애플리케이션 Lifecycle을 보여주는 그림으로 Master Server에서의 애플리케이션 상태의 진행과정이다.

INSTALLED → DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING



도메인에서의 애플리케이션 Lifecycle

다음은 각 상태에 대한 자세한 설명이다.

상태	설명
INSTALLED	애플리케이션 파일이 도메인으로 upload된 상태를 의미한다. 이 상태에서 Deploy 대상을 지정해서 deploy 나 distribute 명령어를 수행할 수 있다.
DISTRIBUTED	애플리케이션이 Distribute를 성공적으로 완료한 상태를 의미한다. 이때 대상 서버들이 모두 SHUTDOWN 상태이더라도 DISTRIBUTED 상태로 나타난다.

상태	설명
RUNNING	애플리케이션이 서비스되고 있는 상태를 의미한다. 애플리케이션에 Deploy 대상으로 설정한 서버들 중 한 서버에서라도 RUNNING 상태로 서비스되고 있으면 도메인에서 이 애플리케이션의 상태는 RUNNING이 된다.



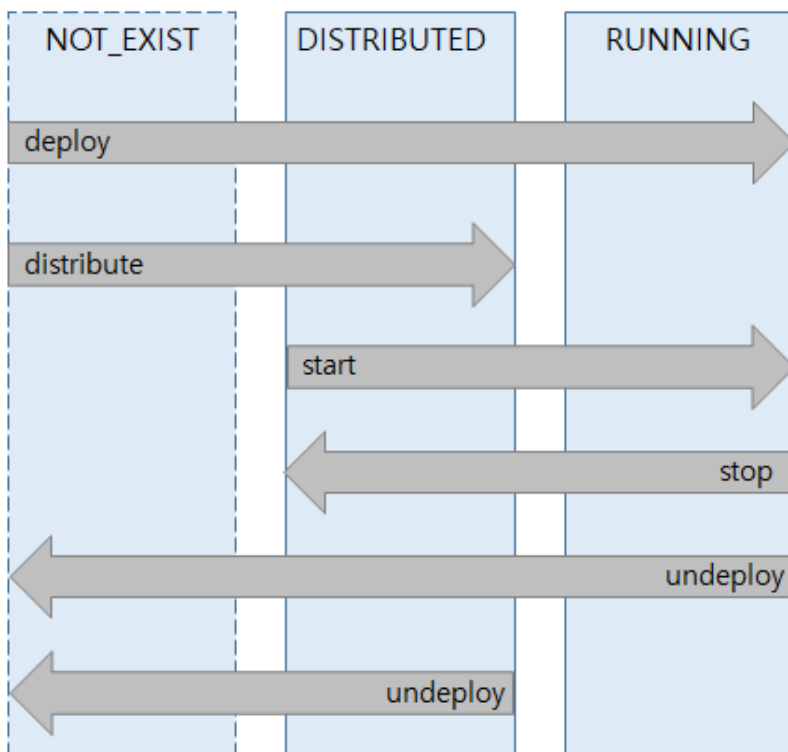
도메인에서는 위의 세 가지 상태 외에 **DEPLOYED**라는 상태가 존재한다.

DEPLOYED 상태란 deploy된 이력이 있는 애플리케이션의 대상 서버가 모두 RUNNING 상태가 아닐 때 도메인에서는 이 애플리케이션의 상태를 DEPLOYED로 나타낸다. 대상 서버들이 running되지 않았다는 것은 애플리케이션을 서비스하고 있는 서버들이 없다는 의미이므로 RUNNING으로 표현하지 않고 DEPLOYED라는 상태로 나타낸다.

서버에서의 애플리케이션 상태

다음은 서버에서의 애플리케이션 Lifecycle을 보여주는 그림으로 서버에서의 애플리케이션 상태의 진행과정이다.

DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING



서버에서의 애플리케이션 Lifecycle

다음은 각 상태에 대한 설명이다.

상태	설명
DISTRIBUTED	Master Server로부터 온 distribute 명령어가 성공한 상태를 의미한다. 애플리케이션은 이미 서비스될 준비를 마쳤고 start 명령어를 통해 애플리케이션을 서비스 가능한 상태로 만들 수 있다.
RUNNING	애플리케이션이 해당 서버에서 서비스되고 있는 상태를 의미한다.

1.1.4. 애플리케이션 관리 디렉터리 구조

애플리케이션을 deploy할 때 사용되는 디렉터리 구조는 다음과 같다.

도메인 **INSTALL_HOME** 디렉터리

애플리케이션이 도메인에 install되면 **APPLICATION_INSTALL_HOME**에 위치하게 된다.

```
APPLICATION_INSTALL_HOME
|--application_id
|   |--[J]application file(.jar, .war, .ear, .rar)
```

* Legend

- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor

APPLICATION_INSTALL_HOME

Master Server가 존재하는 머신에만 생성되는 디렉터리이다. 애플리케이션을 도메인에 install하면 이 디렉터리에 저장된다. install할 때 설정한 애플리케이션 ID로 디렉터를 생성하고 그 하위에 실제 애플리케이션 파일을 위치시킨다. JEUS에서는 이 디렉터를 **APPLICATION_INSTALL_HOME**이라고 하고 이를 줄여서 **INSTALL_HOME**이라고 한다. **APPLICATION_INSTALL_HOME**은 **DOMAIN_HOME/.applications**를 의미한다.

도메인에서 관리하는 애플리케이션 저장소

도메인에 추가한 애플리케이션 저장소(repository)는 다음과 같은 구조로 구성한다.

```
application repository
|--application file (exploded module)
|   |--[J]application file(.jar, .war, .ear, .rar)
```

* Legend

- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file

- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor

애플리케이션 저장소에는 디렉터리 형태의 애플리케이션과 Archive 형태의 애플리케이션이 모두 저장된다. 이 경우 애플리케이션 ID는 사용자가 부여할 수 없고 항상 파일 이름을 사용하게 된다. 사용자는 애플리케이션 저장소를 추가하고 삭제할 수 있다. 추가 및 삭제 방법에 대해서는 [애플리케이션 저장소 추가/삭제/조희](#)를 참고한다.

Deploy 이미지 디렉터리

서버에 애플리케이션을 deploy할 때 Master Server로부터 애플리케이션을 전송받아서 다음의 경로에 위치시킨다.

```
SERVER_HOME/.workspace/deployed
```

이 위치를 **DEPLOYED_HOME**이라고 한다. DEPLOYED_HOME에 애플리케이션 ID로 디렉터를 생성하고 전송받은 애플리케이션 파일을 하위에 위치시킨다.

또한 Deploy를 진행하면서 archive된 애플리케이션의 경우 파일의 압축을 푼다. 이를 애플리케이션의 **Deploy 이미지 디렉터리**라고 한다. 애플리케이션의 압축을 푸는 디렉터리는 애플리케이션 파일 이름을 변경해서 사용한다. myApp.ear의 경우 압축을 푸는 이미지 디렉터리는 'myApp_ear__'이 된다.

ear 애플리케이션일 경우는 ear 이미지 디렉터리 하위에 모듈의 압축을 푼다. 모듈의 압축을 푸는 디렉터리 이름은 ear의 압축을 푼 디렉터를 생성한 규칙과 동일하게 ejb.jar인 경우는 'ejb_jar'가 된다.

다음은 서버에 deploy된 ear 애플리케이션의 이미지 디렉터리에 대한 설명이다.

```
DEPLOYED_HOME
|--_generated_
|--myApp
  |--myApp_ear__
    |--appclient_jar__
    |--ejb_jar__
    |--lib
    |--META-INF
    |--web_war__
    |--[J]appclient.jar
    |--[J]ejb.jar
    |--[J]web.war
```

* Legend

- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor

애플리케이션의 gen 디렉터리

애플리케이션이 서버에 deploy되면서 서비스에 필요한 파일들을 생성하는 경우가 있다.

EJB 애플리케이션과 웹 애플리케이션의 경우 파일 생성이 필요하다. 이 파일들을 DEPLOYED_HOME에 '_generated_'라는 디렉터리 하위에 생성한다. '_generated_' 디렉터리 하위에 애플리케이션 ID로 디렉터리를 생성하여 각 애플리케이션마다 개별적인 generated 디렉터리를 갖게된다. 이 디렉터리를 애플리케이션의 **gen 디렉터리**라고 한다.

각 애플리케이션의 gen 디렉터리에는 다음과 같은 파일이 생성된다.

- EJB의 경우는 EJB 2.x 형태의 Bean을 Dynamic Proxy 방식이 아닌 Stub 방식으로 사용할 때 Stub 파일과 Skel 파일을 생성한다.
- 웹의 경우는 JSP를 Java로 generate한 파일과 Embedable EJB(Java EE 6부터 웹 애플리케이션 안에 내장된 EJB 모듈을 지원한다. 이를 Embedable EJB라고 한다)를 위한 stub 파일을 생성한다.
- '_appdat.ser'은 애플리케이션의 Deploy 시간을 저장한 serializable 파일이다. 이 파일의 용도는 서버를 shutdown했다가 재부팅할 경우 애플리케이션이 변경되지 않았으면 Deploy 이미지 파일과 generate한 파일을 재생성하지 않도록 해서 부팅 속도를 빠르게 하기 위한 것이다.

다음은 서버에 deploy된 ear 애플리케이션 gen 디렉터리 구조이다.

```
DEPLOYED_HOME
|--_generated_
|--myApp
    |--ejb_jar
    |   |--classes
    |--web_war
    |   |--__embedded_ejb
    |   |   |--classes
    |   |--__jspwork
    |   |--__ws
    |--_appdat.ser
```

* Legend

- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor

1.1.5. 애플리케이션 대상

Deploy를 할 때는 어떤 서버에서 서비스할 것인지를 대상(target)을 명확히 해야 한다. Deploy 대상이 될 수 있는 것은 서버와 클러스터이다. 웹 모듈의 경우 특정 호스트로만 서비스하려면 추가적으로 가상 호스트를 설정하여 deploy할 수 있다.

서버

서버는 애플리케이션을 서비스하는 최소 단위이기 때문에 Deploy의 대상이 될 수 있다. 여러 서버를 대상으로 deploy 명령을 한 경우 지정한 서버 중 하나 이상의 서버에서 Deploy에 실패하거나 하나 이상의 서버가 다운되어 있는 경우 Distribute 단계에서 실패한다. 이 경우 Deploy에 실패한 서버에 성공할 수 있도록 적절한 조치를 취한 후 다시 deploy 명령을 수행해야 한다.

또는 실패한 서버를 Deploy 대상에서 제외하고 다시 deploy 명령을 수행하면 Deploy는 성공할 것이다. 만약 서버가 STANDBY 상태이거나 SUSPENDED 상태라면 사용자가 이 서버를 대상으로 deploy 명령을 해도 서버가 서비스될 수 없는 상태이기 때문에 애플리케이션을 start하지 않는다.

Distribute 작업까지만 진행하고 서버가 다시 start 또는 resume되어서 서비스할 수 있는 RUNNING 상태가 되었을 때 해당 애플리케이션을 start한다. 서버 상태에 대한 자세한 설명은 JEUS Server 안내서의 "Managed Server의 Lifecycle"을 참고한다.

사용자가 서버에 deploy 명령이 아닌 distribute 명령만 수행한 경우에 애플리케이션은 서비스되지 않는다. 서버가 재부팅된 경우에도 이 상태는 유지되어야 한다. 따라서 Master Server에서는 애플리케이션의 상태를 INSTALL_HOME 하위에 ".deployInfo.ser"라는 파일에 기록했다가 서버가 부팅할 때 애플리케이션에 대한 상태를 알려주어서 사용자가 이전에 한 작업을 계속 유지할 수 있도록 해준다.



사용자가 설정 파일을 수동으로 수정한 경우라면 이 상태가 유지되지 않을 수도 있다. 따라서 설정 파일 수정 시에 JEUS에서 제공하는 관리 툴(jeusadmin 등)을 사용할 것을 권장한다.

클러스터

클러스터에 속한 서버에는 서버를 대상으로 deploy할 수 없고 오직 클러스터만 대상으로 deploy할 수 있다. 클러스터를 대상으로 deploy한 경우 클러스터 멤버에 속한 서버 중 하나의 서버에서라도 Distribute가 실패하면 클러스터 Deploy는 실패한다. 하지만 서버 리스트를 대상으로 설정하고 deploy하는 경우와는 달리, 클러스터를 대상으로 deploy하는 경우는 클러스터 멤버 중 일부 서버가 Alive 상태가 아니더라도 Deploy에 성공할 수 있다. 클러스터 멤버 중 Alive 서버 모두에 deploy를 성공하면 클러스터 Deploy는 성공한다.



클러스터는 단순 싱글 서버의 리스트를 의미하는 것이 아니고 서비스를 수행하는 하나의 그룹을 의미한다. 클러스터 멤버 중 Alive 서버에 Deploy를 성공하면 alive 상태가 아닌 서버에도 Deploy가 성공할 가능성이 매우 높아진다. 클러스터를 deploy할 경우 Alive 상태가 아닌 다른 멤버들은 정상화되거나 재기동될 때 또는 새로운 멤버가 클러스터에 추가되어서 부팅될 때 이미 클러스터 deploy되어 있는 애플리케이션을 deploy해야 한다.

클러스터 멤버인데 애플리케이션 대상이 서버로 되어 있는 경우는 deploy하지 않는다. 사용자에게 해당 애플리케이션을 undeploy나 remove-application-target 명령어를 수행할 수 있도록 다음과 같은 WARNING 메시지만 출력한다.

클러스터 멤버인 서버로 지정되어 있는 경우 MS를 부팅할 때 발생하는 로그 메시지

```
WARNING: The server[server1] is part of the cluster[cluster1], so this application[examples] can only
be deployed to the cluster.
Deploy again this application to the cluster target.
```


deploy 명령으로 이런 문제는 발생할 수 없다. 클러스터 멤버인 서버를 별개의 서버 target으로 deploy를 시도하는 경우에는 deploy 명령이 수행되지 않는다. **이런 문제가 발생할 수 있는 경우는 설정 파일을 수동으로 수정한 경우이다.**

서비스되고 있는 독립된 서버를 클러스터 멤버로 동적 추가한 경우는 서버에서 이미 서비스되고 있는 애플리케이션을 undeploy하고 애플리케이션 대상에서도 해당 서버를 삭제한다. 또한 클러스터를 대상으로 deploy해야 하는 애플리케이션들도 동적으로 deploy한다. 이미 서버를 대상으로 deploy되어서 서비스하고 있는 애플리케이션이기는 하지만 클러스터에 추가하고 새로운 서비스를 하겠다는 사용자의 의도이기 때문에 기존 애플리케이션을 undeploy하고 새로운 애플리케이션을 deploy해도 무방하다.

클러스터에 동적으로 새로운 멤버를 추가하는 것이 가능하기 때문에 애플리케이션에 대한 처리도 자동으로 해주고 있다. 하지만 이렇게 싱글 서버와 클러스터를 이미 별도로 구성해서 서비스하다가 싱글 서버를 클러스터의 멤버로 동적추가하는 경우는 자주 사용하지 않고, 사용자 의도이긴하나 서버에서 이미 진행 중인 서비스에 문제가 될 수도 있다. 따라서 클러스터를 확장하는 경우에는 새로운 서버를 추가하고 이 서버를 클러스터의 멤버로 추가하는 것을 권장한다.

가상 호스트(Virtual Host)

웹 엔진에서는 가상 호스트를 이용해 웹 컨텍스트를 특정 호스트에서 서비스할 수 있도록 하고 있다. 가상 호스트는 단독으로 지정할 수 없고 대상이 되는 서버 또는 대상이 되는 클러스터와 함께 지정할 수 있다. 대상이 되는 가상 호스트 없이 deploy할 경우에는 기본 가상 호스트에 deploy된다.

가상 호스트에 대한 자세한 내용은 JEUS Web Engine 안내서의 "가상 호스트"를 참고한다. deploy 명령에 자세한 설명은 [애플리케이션 Deploy](#)를 참고한다.

1.2. Managed Server(MS)에서의 Deploy

본 절에서는 MS에서 애플리케이션 또는 모듈을 deploy하는 방법에 대해서 설명한다.

1.2.1. Run-time Deploy

Run-time Deploy는 서버가 기동되어 있는 상태에서 애플리케이션 또는 모듈을 deploy하는 것이다. JEUS에서 제공하는 툴을 통해 Master Server에 접속하여 Run-time Deploy를 할 수 있다.

JEUS 7 이후 버전에서는 Run-time Deploy를 수행하면 **영구 Deploy**가 된다. 영구 Deploy는 deploy나 distribute 명령어를 통해 애플리케이션이 Distribute에 성공하면 domain.xml에 그 내용이 저장되어, Master Server나 MS를 재기동하더라도 해당 애플리케이션의 Deploy 정보를 설정 파일에서 읽어 Boot-time Deploy될 수 있도록 애플리케이션 정보를 유지해주는 것을 의미한다.

Deploy는 Master Server를 통해서만 가능하기 때문에 domain.xml에 애플리케이션 정보를 쓰는 주체는 오로지 Master Server뿐이다. MS에서는 Boot-time에 Master Server로부터 설정을 받고 해당 설정에 등록된 애플리케이션 파일을 다운로드받아 XML 파일의 정보를 바탕으로 Boot-time Deploy를 진행할 수 있다. Runtime Deploy의 명령 방법에 대한 자세한 내용은 [애플리케이션 Deploy](#)를 참고한다.

1.2.2. Boot-time Deploy

Boot-time Deploy는 서버가 부팅할 때 domain.xml에 등록된 애플리케이션을 자동으로 deploy하는 것을 의미한다. JEUS 7 이후 버전부터는 기본적으로 deploy한 애플리케이션의 정보가 domain.xml에 기록되기 때문에, 성공적으로 deploy된 이력이 있는 애플리케이션들은 boot-time deploy의 대상이 된다.

DEPENDENT 상태에서의 Boot-time Deploy

MS는 부팅할 때 Master Server로부터 설정을 전송받아온다. 이 설정에 등록된 애플리케이션이 있으면 애플리케이션의 대상을 보고 자신의 서버에 deploy해야 할 애플리케이션인지를 판단하여 Deploy를 진행한다. 이때 각 애플리케이션마다 Deploy 작업을 진행하는 것이 아니라 모든 애플리케이션에 대한 Distribute 작업을 먼저 진행하고, 모든 애플리케이션의 distribute가 성공한 경우에 애플리케이션 Start 작업을 진행한다.



서버가 클러스터에 속한 멤버인데 애플리케이션이 서버 target으로 명시되어 있는 경우 해당 Boot-time에 애플리케이션은 deploy되지 않는다. Deploy의 대상은 클러스터에 속한 서버가 될 수 없다. 클러스터에 속한 서버는 오직 클러스터 단위로만 Deploy가 가능하다.

JEUS 툴을 통해 Deploy를 수행한 것이라면 이는 보장된다. 하지만 수동으로 XML 파일을 수정한 경우에는 JEUS에서 이를 보장해줄 수 없으니 가급적이면 JEUS가 제공하는 툴을 사용할 것을 권장한다.

Boot-time에 서버에서 진행하는 Distribute 작업은 runtime에 서버에서 진행하는 Distribute 작업과 크게 다르지 않다. 먼저 애플리케이션 파일을 Master Server로부터 전송받는다. 만약 서버에서 이전에 전송받아놓은 파일이 존재한다면 서버에 캐시된 애플리케이션 파일과 Master Server에 install된 애플리케이션 파일의 시간을 비교하여 파일 전송 여부를 결정한다. 그 후에 애플리케이션 파일의 압축을 풀고 클래스를 로딩하고 애플리케이션이 서비스될 수 있는 환경을 구성하는 Distribute 작업을 진행한다.

서버를 부팅할 때 하나의 애플리케이션이라도 distribute에 실패하면 서버의 상태는 STANDBY 상태가 되고 그 시점에서 부팅이 멈추게 된다.

Boot-time Deploy할 때 distribute가 완료된 시점에서 MS는 Master Server로부터 해당 애플리케이션이 start를 수행할 수 있는 상태인지를 확인한다.

Master Server에서의 애플리케이션 상태가 DEPLOYED, RUNNING 상태인 애플리케이션에 대해서만 Start 작업을 수행할 수 있다. 만약 Master Server로부터 애플리케이션의 상태를 얻어오는 작업이 실패했다면 서버는 애플리케이션 Start 작업을 진행하지 않는다. 이때 서버는 부팅을 중단하고 STADNBY 상태에 머무르게 된다.

Master Server에서의 애플리케이션 상태에 상관없이 애플리케이션을 start하여 서비스가 가능하려면, distribute에 실패한 애플리케이션이 있을 때와 마찬가지로 start-server 명령에 -force 옵션을 주고 수행하여 서버를 RUNNING 상태로 만들어 서비스 가능하도록 할 수 있다.



Master Server에서 DISTRIBUTED 상태인 애플리케이션은 서버에서도 Distribute 작업까지만 진행할 수 있게 하기 위해서 Master Server에서는 별도의 파일에 이 정보를 기록하여 Master Server가 다운되었다가 부팅된 상황에서도 애플리케이션 상태를 유지해줄 수 있도록 하고 있다.

서버가 SUSPENDED 상태에서 애플리케이션이 stop, start된 경우에도 마찬가지로 서버가 다시 resume이 되서 RUNNING 상태가 되었을 때 애플리케이션의 바뀐 상태를 잘 유지해줄 수 있도록 해야 한다. 이때는 Master Server로부터 애플리케이션의 상태를 다시 받아오지는 않는다. SUSPENDED 상태이더라도 Master Server로부터 애플리케이션 stop, start 명령은 받을 수 있기 때문에 실제 애플리케이션이 MS에서 명령을 수행할 수 없는 상태이더라도 Master Server로부터 온 start, stop 작업을 기억했다가 서버가 다시 resume이 되는 순간 애플리케이션의 상태를 유지해줄 수 있도록 하고 있다.



서버가 SUSPENDED 상태일 때 애플리케이션 stop 명령이 오면 서버에서는 이미 모든 애플리케이션이 stop된 상태이기 때문에 ApplicationAlreadyStoppedException이 발생한다. Master Server에서는 이런 상황에서 발생한 Exception은 애플리케이션 stop 작업이 실패했다고 취급하지 않는다.

서버가 SUSPENDED 상태일 때 애플리케이션 start 명령이 오면 서버에서는 애플리케이션 Start 작업을 수행할 수가 없다. 이때 서버가 suspend되기 이전에 해당 애플리케이션의 상태가 DISTRIBUTED 상태였다면 서버가 resume이 되면서 이 애플리케이션을 start해야 한다.

INDEPENDENT 상태에서의 Boot-time Deploy

서버가 INDEPENDENT 상태로 부팅된 경우는 Master Server의 관리를 받지 않는다. 또한 Master Server와 연결할 수 없는 상황이기 때문에 애플리케이션 파일 전송도 받을 수 없다. 따라서 Boot-time Deploy할 때에도 이전에 받아놓았던 애플리케이션 파일이 없거나 잘못되어 있다면 Deploy를 진행할 수 없다.

INDEPENDENT 상태로 서버를 부팅할 때는 애플리케이션 파일에 접근할 수 있는 경우에만 Deploy를 진행할 수 있다. 애플리케이션 파일을 접근할 수 있는 경우는 다음의 3가지 경우를 말한다.

- MS가 Master Server와 같은 머신에 존재하여 install된 애플리케이션 원본 파일에 접근 가능한 경우
- 애플리케이션 파일이 NAS에 존재하여 MS에서도 접근 가능한 경우
- 압축을 풀어놓은 Deploy 이미지 디렉터리가 남아있는 경우

위의 경우에 해당되지 않는다면 해당 애플리케이션은 deploy해야 할 애플리케이션 파일을 찾을 수 없기 때문에 Deploy에 실패한다. 이 경우에도 마찬가지로 서버는 STANDBY 상태가 되며 부팅을 멈추게 된다.

INDEPENDENT 상태로 서버를 띄울 때에는 DEPENDENT 상태와는 달리 Master Server로부터 distribute만 진행해야 하는 애플리케이션 리스트를 받아올 수 없다. 따라서 distribute에 성공한 애플리케이션 모두를 start시켜 서비스가능한 상태로 만든다.

INDEPENDENT 상태로 부팅한 서버가 Master Server와 연결되어 DEPENDENT 상태가 되면, 옵션에 따라 distribute에 실패한 애플리케이션들을 모두 distribute를 재시도한다. 실패했던 애플리케이션들이 모두 distribute에 성공하면 Boot-time Deploy할 때와 마찬가지로 Master Server로부터 start해야 하는 애플리케이션 리스트를 받아와서 start시켜준다.

또한, distribute만 해야 하는 애플리케이션이 start되어서 서비스되고 있는 경우 이 애플리케이션은 stop시켜준다. 단, 서버가 부팅 중에 Master Server가 failback되어 DEPENDENT 상태가 되었다면 위 작업을 별도로 진행하지는 않는다.

1.2.3. 애플리케이션 동기화

애플리케이션 파일의 동기화는 Master Server의 주요 역할 중 하나이다. Master Server와 MS 사이에 애플리케이션 파일이 동기화되는 경우는 크게 몇 가지가 있다.

- Run-time Deploy를 수행할 경우

MS에서는 새로 deploy할 애플리케이션 파일을 Master Server로부터 전송받는다.

- Boot-time Deploy를 수행할 경우

MS에 deploy해야 할 애플리케이션 파일이 변경되었다면 MS에서는 Master Server로부터 애플리케이션 파일을 새로 전송받는다.

- MS가 INDEPENDENT 상태였다가 DEPENDENT 상태로 변환된 경우

MS가 INDEPENDENT 상태였다가 DEPENDENT 상태로 변환되어 Master Server의 관리를 받게 된 경우, 옵션을 통해서 애플리케이션의 동기화를 진행한다. 도메인 설정 중 '**enable-to-resynchronize-applications**'를 true로 설정한 경우에 한해서 동기화를 진행하고 기본값은 false로 동기화를 진행하지 않는다. 해당 옵션이 true로 설정 된 경우, MS에서는 deploy에 실패한 애플리케이션에 대해서 애플리케이션 파일을 다시 전송받는다. 또한 Deploy에 성공한 애플리케이션 중에서 Master Server에서 관리하는 애플리케이션 파일이 변경된 경우에도 MS에서 애플리케이션 파일을 다시 전송받는다.

Master Server에서는 애플리케이션 파일의 동기화뿐만 아니라 애플리케이션의 도메인에서의 상태도 동기화한다.

Boot-time Deploy를 수행할 때 애플리케이션이 Master Server에서 DISTRIBUTED 상태로 관리되고 있다면 DISTRIBUTED 상태까지만 만들어준다. 이 애플리케이션을 서비스하기 위해서는 사용자가 **start-application** 명령어를 수행해야 한다. 뿐만 아니라 MS가 INDEPENDENT 상태였다가 DEPENDENT 상태로 변환된 경우 MS에서는 애플리케이션 파일뿐만 아니라 상태도 동기화해준다. DEPENDENT 상태가 된 애플리케이션은 Master Server에서 관리하고 있는 애플리케이션의 상태에 따라 MS에서 서비스되고 있는 애플리케이션의 상태도 바뀌게 된다.

INDEPENDENT 상태의 MS에서는 모든 애플리케이션이 서비스될 수 있도록 RUNNING 상태로 만들었지만, Master Server와 연결되어 DEPENDENT 상태가 된 후에는 Master Server의 상태를 따른다.

Master Server에서 애플리케이션이 DISTRIBUTED 상태였다면 MS의 애플리케이션도 중지하여 DISTRIBUTED 상태로 만든다. 또한 Master Server에서 이미 undeploy한 애플리케이션이 MS에 존재한다면 이 또한 MS에서도 undeploy한다.

1.3. Boot-time Auto Deployment

서버 기동 시 지정한 경로를 탐색하여 해당 경로에 위치한 애플리케이션 파일을 자동으로 배포해 주는 기능이다. 자동으로 배포한 애플리케이션은 일반 애플리케이션과 달리, 다음과 같은 특징을 갖는다.

- 애플리케이션 ID는 확장자를 제외한 파일명으로 자동 설정된다. 만약 설정에 이미 동일한 ID를 갖는 애플리케이션이 존재하는 경우 서버 기동 중 오류가 발생한다.
- 배포한 애플리케이션에 대한 내용이 domain.xml에 저장되지 않는다. 따라서 자동 배포한 애플리케이션에 대한 모든 동작은 런타임에만 적용된다.

- 애플리케이션 파일을 마스터 서버가 중앙에서 관리하지 않기 때문에 애플리케이션 파일 동기화 작업 등이 일어나지 않는다.
- 클러스터 타겟 개념이 존재하지 않는다. 자동 배포한 애플리케이션은 배포 대상 서버가 클러스터에 속해 있더라도 항상 서버를 대상으로 배포한 것으로 간주한다. 이는 애플리케이션 파일을 중앙 관리하지 않기 때문에 클러스터 전체에 애플리케이션을 동기화하거나 하는 작업을 수행하는 것이 곤란하기 때문이다.
- 일반 애플리케이션과 달리, 자동 배포한 애플리케이션의 경우 배포 대상 서버가 모두 종료된 경우 마스터 서버에서 해당 애플리케이션이 undeploy된 것으로 간주하여 정보를 삭제한다.

자동 배포할 애플리케이션을 위치시킬 경로 및 기능 사용 여부는 서버 별로 시스템 프로퍼티를 JVM Option에 지정하여 설정할 수 있다.

- `jeus.server.enable.auto-deploy` 시스템 프로퍼티를 사용하여 기능 사용 여부를 설정 가능하다. 기본값은 `true`(사용함)이다.
- `jeus.server.auto-deploy.dir` 시스템 프로퍼티를 사용하여 자동 배포 애플리케이션을 위치시킬 경로를 지정할 수 있다. 기본값은 `DOMAIN_HOME/auto-deploy`이다.

1.4. Auto Redeploy

JEUS 6까지는 지정된 디렉터리 또는 애플리케이션을 주기적으로 검사하면서 변경된 내용이 있을 경우 자동으로 Deploy 시도가 가능했다. 이 기능을 사용하면 애플리케이션을 수동적으로 deploy할 필요가 없으므로 애플리케이션을 개발할 때나 빈번한 업데이트가 필요한 애플리케이션의 경우 유용하게 사용할 수 있었다. **JEUS 7 이후 버전에서는 JEUS6 이전과 같은 Auto Deploy 기능은 지원하지 않는다. 대신 도메인에서 애플리케이션의 변경을 감지해서 자동으로 redeploy하는 기능을 제공한다.**



마스터 서버에서 애플리케이션을 중앙 관리하는 것을 기본으로 하기 때문에 Auto Deploy를 지원하기가 어려워졌다. 애플리케이션 대상을 도메인 단위로 보기도 어렵고 Exploded 모듈의 경우는 다른 머신으로 전송되지 않기 때문에 Auto Deploy 기능은 제공하지 않는다. 대신 최초 deploy할 때 애플리케이션 대상과 Auto Redeploy를 체크할 주기를 설정해서 수동으로 deploy를 하면 애플리케이션 파일이 변경된 경우에 이를 감지하여 애플리케이션 대상으로 Auto Redeploy를 할 수 있다.

Auto Redeploy 설정을 한 애플리케이션에 대해서는 설정한 주기마다 애플리케이션 파일의 변경을 체크한다.

- Archive 모듈의 경우는 Archive 파일의 last modified time으로 애플리케이션의 변경을 감지해서 redeploy한다.
- Exploded 모듈의 경우는 표준 Deployment Descriptor(이하 DD)의 last modified time으로 변경을 감지해서 애플리케이션을 redeploy한다.

Redeploy는 Master Server가 대상이 되는 서버로 redeploy 명령을 보내고, 서버에서는 Auto Redeploy와 일반 redeploy 명령이 다르게 동작하지 않는다. Master Server에서는 모든 서버에 Redeploy가 성공해야 전체 성공으로 간주한다. 파일이 변경되었다고 해서 바로 redeploy되지 않을 수 있다.

설정된 주기가 되어야 애플리케이션 파일 변경 여부를 체크하기 때문에 최대 설정한 주기만큼 기다려야 Redeploy가 진행된다. 이 주기의 기본값은 10초이다.

1.5. 디렉터리 모드의 Deploy

애플리케이션은 Jakarta EE 스펙에서 정의한 각 애플리케이션 타입에 맞는 형태로 패키징되어 있어야 한다. 하지만 개발 단계에서의 편의성을 제공하기 위해 패키징을 하지 않은 형태의 애플리케이션 또한 deploy할 수 있도록 지원하고 있다. JEUS에서는 이런 디렉터리 형태의 애플리케이션을 **Exploded 모듈**이라고 한다.

Master Server와 같은 머신에 존재하는 서버나 클러스터를 대상으로 한 경우나 Master Server와 다른 머신에 있는 서버나 클러스터이더라도 NAS(Network Attached Storage)와 같이 다른 머신에서도 접근 가능한 경로에 위치한 경우에 한해서 Deploy가 가능하다. 이런 애플리케이션은 **install-application** 명령어를 통해서 Master Server의 애플리케이션 INSTALL_HOME으로 install시키는 것이 아니라 애플리케이션의 상위 디렉터리를 '**Application Repository**'로 지정하거나 deploy 명령을 할 때 -path 옵션을 주고 deploy해야 한다. 이런 애플리케이션을 deploy할 때도 옵션으로 애플리케이션의 ID를 줄 수 있다. ID를 주지 않은 경우는 애플리케이션 파일 이름이 애플리케이션의 ID가 된다.



JEUS는 디렉터리 모드로 deploy한 애플리케이션에 대한 동기화나 관리 작업을 수행하지 않는다. 해당 디렉터리는 사용자가 관리해야 한다.

1.6. 애플리케이션 저장소

도메인 환경에서는 해당 도메인에서 서비스되는 모든 애플리케이션을 Master Server에서 관리하기 때문에 애플리케이션을 deploy하기 전에 애플리케이션 파일을 Master Server로 install하는 작업을 항상 선행해야 한다. 하지만 도메인에 애플리케이션 저장소(Application Repositories)를 추가한 경우에는 애플리케이션 파일의 Install 작업을 생략해도 애플리케이션을 deploy할 수 있다.

1.6.1. 애플리케이션 저장소 추가/삭제/조회

도메인에서는 Master Server로 install된 애플리케이션을 한 곳에 모아 관리한다. 이 디렉터리에는 수동으로 애플리케이션을 위치시킬 수는 없다. 여러 저장소(repository)를 설정할 수 있고, 설정한 디렉터리는 모두 애플리케이션 저장소로 인식된다. 애플리케이션 저장소를 추가, 삭제하고 설정되어 있는 정보를 조회하려면 JEUS가 제공하는 툴을 사용해야 한다.

도메인에 애플리케이션 저장소로 설정한 디렉터리에는 Jakarta EE 애플리케이션을 위치시킬 수 있다. Archive 모듈뿐만 아니라 Exploded 모듈도 가능하다.

도메인에 등록된 애플리케이션 저장소(repository)에는 install-application/uninstall-application 명령어를 통해서 애플리케이션 파일을 추가, 삭제하는 것은 불가능하다. 애플리케이션 파일을 수동으로 추가, 삭제하면 도메인에서 이를 자동으로 인식할 수 있다. 이렇게 위치시킨 애플리케이션은 도메인에서 INSTALLED 상태로 인식된다. 삭제된 애플리케이션 파일은 애플리케이션 정보에 나타나지 않고 더 이상 도메인에서 관리하지 않는다.

애플리케이션의 ID는 애플리케이션 파일 이름이기 때문에 deploy 명령을 할 때 파일 이름을 -id 옵션으로 지정한다. 애플리케이션 파일이 업데이트하려면 이 저장소의 파일을 수동으로 업데이트한다. 만약 애플리케이션을 deploy할 때 Auto Redeploy 주기를 설정했다면 Master Server에서 파일의 변경 여부를 자동으로 감지하여 Auto Redeploy가 가능하다.

애플리케이션 저장소를 추가할 때 저장소에 존재하는 파일 이름이 이미 도메인에서 사용되고 있는 애플리케이션

ID라면 해당 애플리케이션은 도메인에서 인식되지 않고 다음과 같은 로그 메시지가 발생한다.

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0005] WARNING:
The application[myApp] already exists on /JEUS_HOME/.applications/myApp.ear.
Change the file name of [/home/user1/apps/myApp].
```

중복된 애플리케이션 ID를 사용하는 경우 애플리케이션 정보 요청하거나 Master Server를 시작하는 경우도 동일한 로그가 발생한다.

또한 추가하려는 애플리케이션 저장소에 유효한 애플리케이션이 존재하지 않는다면 다음과 같은 로그 메시지가 발생한다.

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0062] WARNING:
The repository(/home/user1/apps) does not contain any valid applications,
but new applications can be added.
```

애플리케이션 저장소를 삭제할 때 저장소에 존재하는 파일이 도메인에 distribute되어 있거나 deploy되어 있다면 애플리케이션 저장소는 삭제되지만 애플리케이션이 도메인의 관리 대상에서 삭제되지는 않고 다음과 같은 로그 메시지가 발생한다.

```
[2016.08.08 12:44:04.328][1] [adminServer-93] [Deploy-0124] WARNING:
The application[myApp] is RUNNING in repository[/home/user1/apps].
To remove this application from the domain, undeploy it.
```



애플리케이션 저장소를 삭제하려고 하는 의도가 이미 서비스되고 있는 애플리케이션까지 도메인에서 제외시키겠다는 의도인지 알 수 없다. 서비스되고 있는 애플리케이션을 제외한 다른 애플리케이션만 제외될 것을 원할 수도 있기 때문에 JEUS에서 이를 판단해주기가 모호하다. 따라서 저장소를 삭제할 때 서비스 중인 애플리케이션을 내리고 싶다면 undeploy 명령어를 수행해야 한다.

콘솔 툴 사용

도메인에 애플리케이션 저장소를 추가하려면 **add-application-repository** 명령어를 사용하고, 삭제하려면 **remove-application-repository** 명령어를 사용한다. install할 때 설정하는 ID로 애플리케이션 관리 디렉터리 하위에 디렉터리를 생성하고 거기에 애플리케이션 Archive 파일을 위치시킨다.

다음은 콘솔 툴으로 애플리케이션 저장소를 추가, 삭제, 조회하는 예제이다.

```
[MASTER]domain1.adminServer>add-application-repository /apps/test
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or list-application-repositories"

[MASTER]domain1.adminServer>list-application-repositories
Application Repositories
=====
```



```

+-----+
|               Path to Application Repository               |
+-----+
| /apps/test |
+-----+
=====

[MASTER]domain1.adminServer>remove-application-repository /apps/test
Successfully performed the REMOVE operation for An application repository.
Check the results using "remove-application-repository or list-application-repositories"

[MASTER]domain1.adminServer>list-application-repositories
Application Repositories
=====
+-----+
|               Path to Application Repository               |
+-----+
(No data available)
=====

```



콘솔 툴에서 애플리케이션 저장소를 추가, 삭제, 조회하는 명령어에 대한 자세한 설명은 JEUS Reference 안내서의 "add-application-repository"와 "remove-application-repository", 그리고 "list-application-repositories"를 참고한다.

1.6.2. 애플리케이션 저장소에 있는 애플리케이션 Deploy

애플리케이션 저장소를 추가한 후에는 저장소에 존재하는 애플리케이션을 deploy할 수 있다. 자세한 내용은 [애플리케이션 저장소에 있는 애플리케이션 Deploy](#)을 참고한다.

1.7. path를 지정하여 Deploy

도메인 환경에서 권장하는 애플리케이션 파일의 관리 방법은 2가지로 구분된다.

애플리케이션 파일이 도메인에 의해 관리되길 원한다면 애플리케이션을 INSTALL_HOME으로 install하는 것을 권장하고 사용자가 직접 애플리케이션 파일을 관리하려면 애플리케이션 저장소를 설정하는 것을 권장한다. 애플리케이션 파일을 직접 관리해도 실제 운영환경에서는 도메인에서 사용할 애플리케이션 파일을 업무별로 묶어 몇 개의 디렉터리에 위치시키는 것을 권장하기 때문이다. 하지만 개발 단계에서는 이렇게 애플리케이션을 모아두지 않고 별개로 각각 다른 경로에 존재할 수도 있다. 이 경우 개발의 편의성을 위해 애플리케이션 deploy할 때 **-path** 옵션을 주고 deploy할 수 있도록 제공한다.

path를 지정해서 애플리케이션을 deploy할 때는 -id 옵션은 설정할 수 없다. Exploded 모듈이나 Archive 모듈 모두 Deploy가 가능하고, 애플리케이션 파일의 경로를 -path 옵션으로 주고 deploy한 애플리케이션은 도메인의 애플리케이션 INSTALL_HOME으로 복사하는 과정 없이 애플리케이션을 Master Server에 등록하여 Deploy를 진행한다. 이때 애플리케이션 ID는 애플리케이션 파일 이름을 그대로 사용한다.

path를 지정하여 deploy하는 방법에 대한 자세한 내용은 [path를 지정하여 Deploy](#)을 참고한다.

1.8. Staging Mode Deploy

Exploded 모듈의 경우는 Master Server와 같은 머신에 존재하는 MS에만 deploy할 수 있다. Exploded 모듈의 경우는 주로 개발 단계에서 사용하므로 여러 머신을 사용하지 않기 때문에 같은 머신에서만 Exploded 모듈을 deploy하고 서비스할 수 있다. 하지만 테스트 단계에서는 애플리케이션을 Exploded 모듈 형태로 유지하면서 여러 머신에 도메인을 구성하여 테스트할 수 있기 때문에 Exploded 모듈을 다른 머신에 deploy할 수 있는 방법이 필요하다.

애플리케이션을 deploy할 때 **-staging** 옵션을 주고 deploy하면 Exploded 모듈을 압축하여 다른 머신에 존재하는 MS에도 전송해줄 수 있다. staging으로 deploy된 애플리케이션은 MS에서는 Archive 모듈과 동일하게 취급된다. 압축을 풀어서 deploy하고, MS가 재기동될 때는 Master Server와 애플리케이션 파일을 동기화한다.



Staging 모드 애플리케이션에서 MS와 Master Server 사이에 동기화 대상이 되는 파일은 원본 애플리케이션 디렉터리가 아니라, Master Server가 원본 애플리케이션 디렉터리를 압축한 압축 파일이다. 따라서, 단순히 원본 디렉터리에 있는 파일만 수정한 경우에는 MS 재기동할 때 동기화가 일어나지 않는다.

애플리케이션의 내용이 변경되었다면 redeploy 명령으로 변경된 파일을 MS에 적용하고 다시 deploy할 수 있다. redeploy 명령어를 실행할 때는 -staging 옵션을 줄 필요없이 기존 Archive 모듈을 redeploy할 때와 마찬가지로 redeploy 명령어를 수행한다. 단, 이때는 -path 옵션을 주면 안 된다. MS에서는 redeploy 명령어를 받아서 우선적으로 애플리케이션 파일에 대한 동기화 작업을 진행하고 기존 애플리케이션을 undeploy한 뒤 새로운 애플리케이션을 deploy한다.

Staging 모드로 애플리케이션을 deploy하는 방법에 대한 자세한 내용은 [Staging Mode Deploy](#)를 참고한다.

1.9. 애플리케이션 Undeploy

JEUS 관리 도구를 이용해서 명시적으로 애플리케이션을 undeploy하는 것과 서버 셧다운 시점에 애플리케이션이 undeploy되는 것은 서로 동작 방식이 구분된다.

다음은 두 동작의 차이점이다.

	관리 명령어에 의한 Undeploy	서버 셧다운 시점의 Undeploy
파일 삭제 여부	서버 내부적으로 생성한 파일을 삭제한다. 예를 들어 JSP, EJB 컴파일 결과물들이 삭제되고 EJB 3.0 스케줄 정보가 DB에서 삭제된다.	서버 내부적으로 생성한 파일을 삭제하지 않는다.
적용되는 Timeout	Graceful Undeploy Timeout	Graceful Shutdown Timeout(Undeploy Timeout은 적용되지 않는다.)

2. Graceful Undeployment와 Redeployment

본 장에서는 애플리케이션을 undeploy하고 redeploy할 때 이미 처리 중인 요청을 모두 처리해주고 undeploy나 redeploy 작업을 진행하도록 하는 Graceful Undeploy와 Graceful Redeploy에 대해 설명한다.

2.1. Graceful Undeployment

Graceful Undeploy는 애플리케이션에 undeploy 명령이 들어왔을 때 이미 서비스 중인 사용자 요청을 모두 처리한 후 undeploy가 진행되는 것을 의미한다.

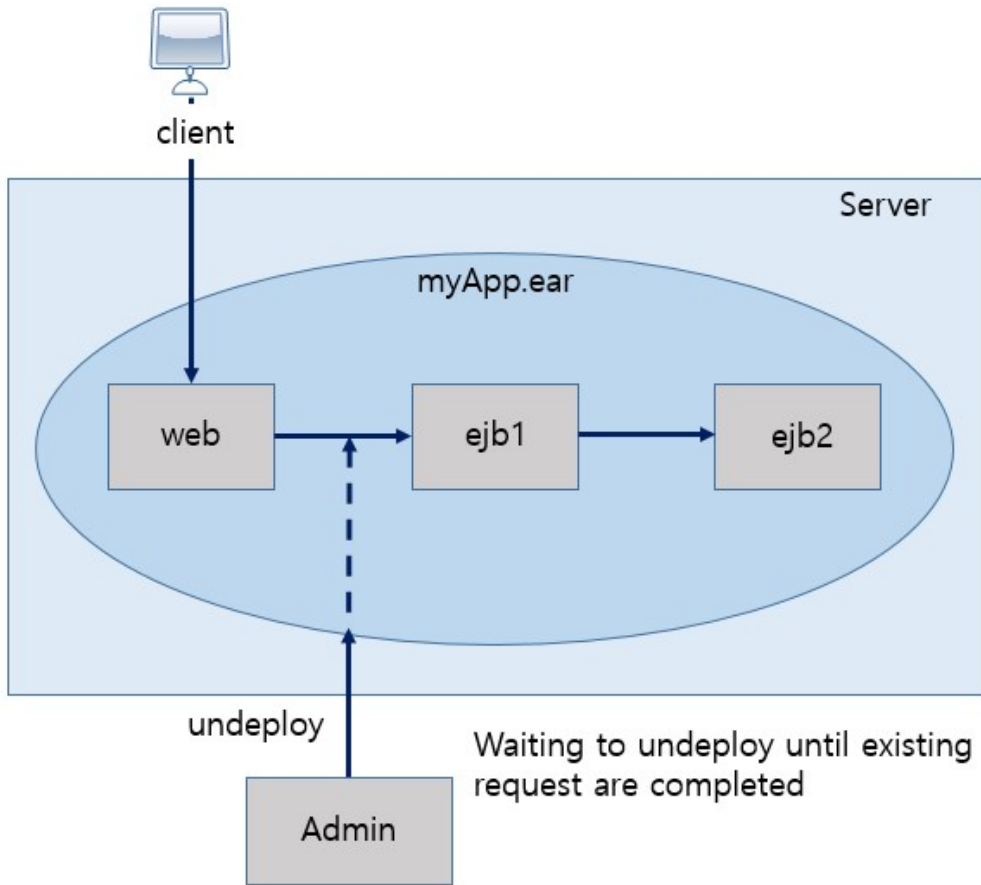
EAR, EJB, 웹 애플리케이션은 undeploy 명령이 들어왔을 때 사용자 요청 처리를 하고 undeploy를 진행하는 Graceful Undeploy로 진행한다. 단, EJB의 경우는 SessionBean만 Graceful Undeploy가 가능하다.

애플리케이션을 undeploy할 때 사용자로부터 온 요청 처리가 아직 미완료 상태라면, 해당 요청들이 모두 완료될 때까지 대기한 후에 undeploy를 수행한다. undeploy 명령어를 실행할 때 -gracefultimeout을 옵션으로 줄 수 있고, 여기에 설정한 값에 따라 요청이 완료되기 전에 undeploy가 수행될 수도 있다. 아직 요청이 완료되지 않은 상태에서 Graceful Timeout으로 undeploy를 해야 한다면 아직 처리 중인 남아있는 요청 Thread에 인터럽트 신호를 보낸다.

undeploy 명령이 들어오면 애플리케이션은 서비스를 할 수 없는 상태인 DISTRIBUTED로 바뀌게 된다. 그리고 이미 처리 중인 요청이 완료될 때까지 또는 사용자가 준 Graceful Timeout이 될 때까지 기다린다. Graceful Undeploy를 진행하려면 사용자가 undeploy 명령어를 실행할 때 적절한 -gracefultimeout을 옵션으로 주어야 한다. gracefultimeout 옵션 값을 주지 않은 경우는 5분으로 동작한다. 5분 동안만 이미 처리 중인 요청의 완료를 보장해주기 때문에 적절한 값으로 설정해야 한다.

애플리케이션이 DISTRIBUTED 상태이면 외부로부터 받은 요청들은 처리될 수 없다. 하지만 EAR의 경우는 undeploy 명령이 시작된 후에도 애플리케이션의 내부 요청은 수행될 수 있다. Standalone EJB의 경우에도 Bean이 여러 개라면 EJB의 undeploy 작업이 시작되어도 Bean 간의 요청은 처리될 수 있다.

다음은 어떤 상황에서 요청이 처리될 수 있는지를 나타낸 그림이다.



EAR 애플리케이션의 Graceful Undeploy

위 그림에서 web에서 ejb1을 호출할 때에는 애플리케이션의 상태가 RUNNING이기 때문에 요청이 처리되는 데 아무런 문제가 없다. 하지만 ejb1이 ejb2를 호출하는 시점에는 이미 애플리케이션의 상태가 DISTRIBUTED로 바뀌었기 때문에 요청을 하더라도 해당 요청이 거부되어야 맞는 상황이라 생각할 수 있다. 하지만 이는 외부 애플리케이션에서 요청을 한 것이 아니고, 같은 EAR 애플리케이션 안에서의 요청이기 때문에 이미 애플리케이션이 DISTRIBUTED 상태가 되었더라도 요청이 처리되는 것을 보장받아야 한다.

위 그림에서처럼 web에서 ejb1을 호출하고, web의 호출을 받은 ejb1에서 ejb2를 호출하는 Call Stack은 하나의 요청이다. 다시 말해 EAR 애플리케이션의 관점에서 보면 ejb1에서 ejb2로의 요청은 새로운 요청이 아니라 web으로부터 시작한 이미 진행 중인 요청인 것이다. EAR 애플리케이션 내부가 아니라 외부 애플리케이션에서 이 애플리케이션의 ejb를 요청하는 경우라면 애플리케이션의 상태에 따라서 요청이 거부될 것이다. 하지만 EAR은 하나의 애플리케이션이기 때문에 EAR이 DISTRIBUTED 상태가 되었다고 해서 EAR 내부 요청인 ejb2에 대한 요청이 실패하면 안 된다. ejb2에 대한 요청은 계속 서비스되어야 하고 이 요청이 완료한 후에 undeploy될 수 있도록 보장한다. 요청이 오래 걸릴 수도 있기 때문에 undeploy 명령에서 준 Timeout이 실제 요청이 처리되는 시간보다 짧을 경우에는 정해진 시간만큼만 기다리고 Undeploy가 진행된다. 이때 요청 처리 중인 Thread에는 인터럽트 신호를 보내서 진행 중이던 작업을 중지한다.



인터럽트 신호를 보낸다고 모든 작업이 중지되는 것은 아니다. Thread에서 처리 중인 작업에 따라 중지될 수도 있고 무시될 수도 있다. 인터럽트에 걸려 작업이 중지되어도 JEUS에서는 해당 Thread에 대해 특별히 처리하지 않는다. 애플리케이션 요청을 서비스하는 Thread에서 인터럽트가 걸린 것이기 때문에 애플리케이션이 이 상황을 고려하여 후처리를 잘 해주어야 한다. Thread 인터럽트에 대한 자세한 내용은 JEUS Server 안내서의 "Thread 제어"를 참고한다.

하나의 애플리케이션 내부에서의 요청에 대한 Graceful Undeploy는 EAR뿐만 아니라 Standalone EJB 애플리케이션에서도 적용된다. EJB 애플리케이션에 Bean이 여러 개 있다면 이 Bean들 사이의 요청도 같은 애플리케이션 내부의 요청이기 때문에 요청 중에 undeploy 명령이 들어와도 이미 처리 중인 요청은 모두 처리해주어야 한다.

undeploy 명령은 기본적으로 5분의 타임아웃이 적용된다. 5분의 시간 동안 애플리케이션의 요청을 보장해주는 것이 기본동작이다. -gracefultimeout 옵션(-to 옵션과 동일)을 줄 수 있어, 애플리케이션의 요청 보장 시간을 변경할 수 있고, 큰 값을 주어서 Graceful Undeploy를 수행할 수도 있다.

JEUS 6의 기본 동작처럼 처리 중인 요청을 기다리지 않고 undeploy를 진행하려면 -force(-f) 옵션을 사용한다. 이 경우 이미 처리 중인 애플리케이션 요청은 보장받지 못한다.

2.2. Graceful Redeployment

Graceful Redeploy는 현재 서비스 중인 애플리케이션의 중단 없이 새로 redeploy한 애플리케이션으로 자연스럽게 서비스를 제공하는 메커니즘이다. 이 기능은 애플리케이션 타입별로 동작에 차이점을 가지고 있다.

본 절에서는 Graceful Redeploy를 사용하기 위해서 공통적으로 알아야 할 사항에 대해 설명하고 그 뒤에 각 애플리케이션 타입별 동작에 대해 설명한다.

2.2.1. Graceful Redeploy를 사용하기 위한 전제 조건

Graceful Redeploy 사용을 위한 전제 조건은 다음과 같다.

- WAR 파일이나 JAR 파일처럼 패키징된 애플리케이션이어야 한다.
- 디렉터리 형태의 애플리케이션의 경우 old와 new 애플리케이션의 디렉터리가 달라야 한다. 디렉터리 형태의 애플리케이션은 원본 디렉터를 그대로 서비스 이미지로 사용하므로 old와 new 애플리케이션이 동일한 디렉터를 가르킨다면 Graceful Redeploy의 요구 조건 자체를 만족시킬 수 없기 때문이다. 그러나 디렉터리 형태의 애플리케이션이라고 할지라도 Staging Mode Deploy한 경우에는 old와 new 애플리케이션이 동일한 디렉터를 가르킬 수 있다. Staging Mode에서는 원본 디렉터리와는 별도로 old와 new 애플리케이션 각각에 대해서 서비스 이미지를 생성하기 때문이다.
- 애플리케이션 저장소에 있는 애플리케이션에 대해서는 new 애플리케이션을 명시하는 Redeploy가 불가능하다. 애플리케이션 저장소에 있는 애플리케이션은 고정적으로 도메인에 install되어있다는 속성을 가지므로 Redeploy에 의한 new 애플리케이션으로의 교체를 불허하기 때문이다. 때문에 애플리케이션 저장소에 있는 애플리케이션이 대해서는 new 애플리케이션을 명시를 전제하는 Graceful Redeploy 역시 불가능하다.
- 애플리케이션을 패키징할 때 META-INF/MANIFEST.MF 파일에 다음과 같은 필드를 추가해야 한다.

또한 디렉터리 형태의 애플리케이션에도 META-INF/MANIFEST.MF 파일을 추가하고 이 필드를 추가해야 한다.

```
Jeus-Use-Graceful-Redeploy: true
```



MANIFEST.MF 파일 규칙상 파일 맨 아랫줄에는 new line(\n)이 있어야 한다.

- Redeploy 대상이 되는 애플리케이션 역시 위의 필드가 있어야 한다. 그렇지 않을 경우 일반적인 Redeploy가 수행된다.
- 디렉터리 형태의 애플리케이션의 경우 old와 new 애플리케이션의 표준 DD에 application-name과 module-name을 동일하게 설정해야 한다.
- 패키징된 파일의 last modified time이 서로 달라야 한다.
- EJB의 경우 Session Bean(remote interface)에 대해서만 Graceful Redeploy가 지원된다.
- SHARED 모드의 애플리케이션의 경우 Graceful Redeploy를 지원하지 않는다. 이 경우 EAR로 같이 묶어서 사용할 것을 권장한다.

2.2.2. Graceful Redeploy에 대한 필수 고려 사항

Graceful Redeploy를 수행할 때 다음의 사항을 반드시 고려해야 한다.

- 서비스 시작 단계 전에 Graceful Redeploy 기능의 사용 여부를 결정해서 전제 조건에서 설명한 바와 같이 애플리케이션을 준비해야 한다.
- 기존 애플리케이션을 바로 undeploy하는 것이 아니고 일정 시간 동안 두 애플리케이션이 동시에 JVM에 존재한다. 따라서 JVM permgen 영역 크기에 대해서 미리 산정해야 한다. 충분한 permgen 영역이 확보되지 않을 경우 OutOfMemoryError가 발생할 수 있다.
- JNI 라이브러리를 사용하는 경우 동시에 같은 라이브러리를 JVM에서 System.loadLibrary()로 로딩할 수 없다. 그러므로 JNI 라이브러리를 사용하는 경우에는 Graceful Redeploy를 사용할 수 없다.

2.2.3. Graceful Redeploy 사용 방법

Graceful Redeploy는 redeploy 명령으로 사용 가능하다. 사용 방법은 실제 서비스 중에 발생할 수 있는 상황을 예로 들어서 설명한다.

먼저 다음과 같이 패키징된 WAR 파일들이 있다고 가정한다.

- old application : myservice.war (08/09/2016 1:00 pm)
- new application : myservice.war (08/25/2016 11:00 am)
- install id = myservice_war, context path = /myservice

서비스 제공자는 2016년 8월 25일 오전에 지난 9일에 패키징해서 deploy했던 myservice.war에 문제점이 있다는 것을 발견했다. 그래서 이를 수정하여 오전 11시에 새로 패키징을 했다. 하지만 이미 myservice.war를 사용 중인 고객들이 있어서 JEUS를 중단시킬 수 없으므로 Graceful Redeploy를 사용하기로 결정했다.

이러한 상황에서 서비스 제공자는 콘솔 툴을 통해서 다음과 같은 명령어를 수행한다.

```
[MASTER]domain1.adminServer>redeploy myservice_war -path /path_to_new_application/myservice.war
```

redeploy 명령에 특별한 옵션을 주는 것이 아니라 META-INF/MANIFEST.MF 파일의 필드와 패키징 시간으로 JEUS에서 Graceful Redeploy 여부를 판단한다. 만약 2016년 8월 9일 오후 1시에 패키징한 애플리케이션의 META-INF/MANIFEST.MF 파일에 전제 조건에서 설명한 필드가 없거나 false로 지정되어 있다면 Normal

Redeploy를 수행한다.



클라이언트의 요청을 처리하는 데 많은 시간이 소요되는 경우에는 redeploy 명령에 이전 애플리케이션을 정해진 시간 안에 종료할 수 있는 Undeploy Timeout 값을 적절히 설정해서 사용한다.

이전 버전 애플리케이션을 강제로 undeploy하는 방법

Graceful Redeploy가 성공적으로 완료된 시점을 기준으로 새로운 서비스 사용자(고객)는 새로운 myservice.war를 사용하게 된다. 기존 서비스 사용자는 계속 이전 버전 myservice.war를 사용한다. 기존 서비스 사용자와 새로운 서비스 사용자를 구분하는 기본적인 기준은 커넥션(소켓 연결)이다. 웹 애플리케이션의 경우에는 HTTP Session도 그 기준에 포함된다.

Graceful Redeploy의 기본 목적은 기존 서비스 사용자가 서비스 종단을 경험하지 않도록 하는 것이다. 따라서 이전 버전 myservice.war에 접근한 커넥션이나 그것을 통해 생성된 HTTP Session이 모두 사라질 때까지 이전 버전 myservice.war는 계속 undeploy되지 않는다. 만약 HTTP Session Timeout이 너무 길어서 일부 HTTP Session에 의해서 undeploy되지 않고 남아 있을 경우에는 서비스 관리자가 다음과 같이 **undeploy** 명령어를 수행해서 강제로 undeploy할 수 있다.

```
[MASTER]domain1.adminServer>undeploy -old myservice_war
```

이전 버전 애플리케이션을 일정 시간이 지난 뒤 자동으로 undeploy하는 방법

이전 버전 myservice.war가 일정 시간 이상 지나면 자동으로 undeploy되게 할 수도 있다. **redeploy** 명령어를 실행할 때 timeout 옵션을 지정한다. timeout 값의 단위는 초이다.

```
[MASTER]domain1.adminServer>redeploy -path /path_to_new_application/myservice.war -to 5
```

이전 버전 애플리케이션과 새로운 애플리케이션이 동시에 존재할 때 이전 버전 애플리케이션으로 rollback하는 방법

이전 버전 myservice.war가 undeploy되지 않은 상황에서 서비스 제공자가 새로운 myservice.war에 문제점이 있음을 발견하였다. 이때 새로운 myservice.war를 undeploy하면 이전 버전 myservice.war로 rollback되어 정상적으로 서비스된다.

다음과 같이 **undeploy** 명령어를 수행한다.

```
[MASTER]domain1.adminServer>undeploy -new myservice_war
```



새로운 애플리케이션에 다소 심각한 문제점이 있어서 바로 undeploy하는 상황에서 사용한다고 가정하고 새로운 애플리케이션은 즉시 undeploy한다. 따라서 새로운

애플리케이션을 사용하던 사용자는 서비스 중단을 경험하게 된다. 이러한 상황이 발생하길 원치 않는 경우에는 이전 버전 myservice.war가 undeploy된 후 그 myservice.war를 다시 redeploy하면 된다.

새로운 애플리케이션을 distribute만 하는 방법

새로운 myservice.war가 제대로 동작하는지 확인 후 실제로 사용자에게 제공하려면 **redploy** 명령어를 실행할 때 distribute만 하도록 지정한다.

```
[MASTER]domain1.adminServer>redploy myservice_war -path /path_to_new_application/myservice.war -distonly
```

새로운 애플리케이션에 대한 서비스 동작 여부를 미리 확인하는 방법은 각 애플리케이션 타입별로 다르다. 새로운 웹 애플리케이션에 대한 동작에 대한 자세한 설명은 [새로운 웹 애플리케이션에 대한 동작 여부를 미리 확인하는 방법](#)을 참고한다.

동작 확인을 한 뒤에 새로운 애플리케이션을 서비스하려면 **start** 명령어를 실행한다.

```
[MASTER]domain1.adminServer>start -new myservice_war
```

새로운 애플리케이션에 문제가 있어서 undeploy하려면 -new 옵션을 주고 **undeploy** 명령어를 실행한다.

```
[MASTER]domain1.adminServer>undeploy -new myservice_war
```

이전 버전 애플리케이션과 새로운 애플리케이션 모두 undeploy하는 방법

이전 버전 애플리케이션과 새로운 애플리케이션이 동시에 존재하는 경우에는 -old 또는 -new 옵션을 이용해서 어느 애플리케이션을 undeploy할지 명시적으로 설정해야 한다. 만약 두 애플리케이션을 모두 undeploy하려면 다음과 같이 -all 옵션을 지정해서 **undeploy** 명령어를 수행한다.

```
[MASTER]domain1.adminServer>undeploy -all myservice_war
```

이전 애플리케이션을 무조건 교체하는 방법

Graceful Redeploy 대신 무조건 Normal Redeploy를 수행할 경우에는 다음과 같이 **redploy** 명령어를 수행한다.

```
[MASTER]domain1.adminServer>redploy myservice_war -path /path_to_new_application/myservice.war -force
```

현재 애플리케이션이 graceful redeploy 설정이 되어 있는지 확인하는 방법

사용 중인 애플리케이션이 graceful redeploy 기능을 사용 중인지 확인하고자 하는 경우 애플리케이션에 대한 상세 정보를 확인해야 한다. Graceful redeploy 기능을 사용 중인 애플리케이션인 경우 상세 정보에 Packaged Time 항목이 표시된다. 이를 통해 사용 여부 및 현재 실행 중인 애플리케이션을 언제 패키징한 것인지를 확인할 수 있다.

```
[MASTER]domain1.adminServer>appinfo -id myservice_war -server server1
Application information for the server [server1] in the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+
| Appli | Applic | Applic | Sta | Target | Target | Packaged Time | Target |
| cation | ation  | ation  | te  | Servers| Cluste |                | VirtualH |
| ID    | Name   | Type   |     |         | rs     |                | ost      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| myser | myserv | war    | RUN | serve  |        | Tue Nov 14    |          |
| vice_w | ice    |        | NING| r1     |        | 14:06:59 KST 2017 |          |
| ar    |        |        |     |        |        |                |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

2.2.4. 웹 애플리케이션 Graceful Redeploy

기존 웹 컨텍스트에서 제공하는 서비스의 중단 없이 새로운 웹 컨텍스트를 redeploy하는 기능이다. 일반적인 redeploy는 기존 웹 컨텍스트를 undeploy하기 때문에 그 과정에서 서비스 이용자가 웹 브라우저에서 에러 페이지를 보게 될 수 있다. 하지만 Graceful Redeploy는 기존 웹 컨텍스트에 대한 서비스 이용자가 있으면 이를 undeploy하지 않고 그대로 유지한 상태에서 새로운 웹 컨텍스트를 deploy한다. 일시적으로 기존 웹 컨텍스트와 새로운 웹 컨텍스트가 동시에 존재하게 된다.

기존 웹 컨텍스트의 서비스 이용자를 판별하는 기준은 현재 진행 중인 요청 존재 여부와 HTTP Session 존재 여부이다. 현재 진행 중인 요청들의 경우 DB 지연 문제나 오랫동안 수행하는 Async Processing이 아니라면 시간이 오래 걸릴 만한 사항이 없을 것이다. 하지만 HTTP Session의 경우 최소한 유후 timeout만큼 남아있기 때문에 기존 웹 컨텍스트는 그 동안 undeploy되지 않는다. redeploy 수행자는 이런 점을 고려하여 기존 웹 컨텍스트에 대한 Undeploy Timeout을 줄 수 있으며, 명령어를 통해서 강제로 undeploy할 수도 있다.

새로운 웹 컨텍스트가 성공적으로 서비스를 시작하면 새로운 요청은 더 이상 기존 웹 컨텍스트로 전달되지 않는다. 기존 웹 컨텍스트의 서비스 이용자는 Graceful Redeploy가 성공적으로 끝나기 이전에 접속한 클라이언트들이다.



기존 웹 컨텍스트의 서비스 이용자들 중 HTTP Session을 사용하는 경우에는 그 세션이 만료되기 전까지 새로운 웹 컨텍스트를 이용하지 못한다. 이를 방지하고 싶은 경우에는 일반적인 redeploy를 수행하거나, Graceful Redeploy를 수행할 때 기존 웹 컨텍스트에 대한 Undeploy Timeout 등을 설정해서 기존 웹 컨텍스트를 강제로 내리도록 해야 한다. 사용 방법은 [Graceful Redeploy 사용 방법](#)을 참고한다.

새로운 웹 애플리케이션에 대한 동작 여부를 미리 확인하는 방법

JEUS에서는 새로운 웹 애플리케이션을 distribute만 해놓고 제대로 동작하는지 확인할 수 있는 방법을 제공한다. 새로운 웹 애플리케이션을 distribute만 하는 방법은 [새로운 애플리케이션을 distribute만 하는 방법](#)을 참고한다.

예제와 같이 myservice.war의 context path가 '/myservice'일 경우 웹 브라우저를 통해서 새로운 웹 애플리케이션을 distribute한 서버의 기본 포트인 8080으로 HTTP 요청을 보내면 된다. 새로운 myservice.war를 distribute만 수행한 서버의 호스트 이름이 'host1'이고 기본 포트가 '9736'인 경우로 가정한다.

```
http://host1:9736/myservice/
```

2.2.5. EJB 애플리케이션 Graceful Redeploy

EJB 애플리케이션의 Graceful Redeploy는 기존의 EJB 요청에 대한 처리를 보장하면서 새로운 EJB를 redeploy하는 기능이다. Graceful Undeploy와 마찬가지로 Session Bean에 대해서만 기능이 제공된다. 따라서 JAR 형식의 애플리케이션을 Graceful Redeploy하려면 해당 JAR는 Session Bean들로만 구성되어 있어야 한다.

Graceful Redeploy 시점을 기준으로 새로운 클라이언트는 새로운 EJB를 Lookup해서 사용하고, 기존 EJB를 사용하던 클라이언트는 사용자가 설정한 Timeout 동안은 요청 처리를 보장받는다. 또한 기존 EJB에 대한 요청 처리가 없다고 판단되면 Timeout 이전이라도 기존 EJB에 대한 undeploy가 이루어진다.

요청 처리에 대한 판단 기준은 EJB의 종류에 따라 달라진다. Stateless Session Bean의 경우는 현재 진행 중인 요청을 기준으로 하고, Stateful Session Bean의 경우는 세션의 존재 여부로 판단한다. 즉, Stateful Session Bean의 경우에는 모든 세션이 destroy됐을 때 더 이상의 추가 요청이 없을 것이라고 판단하고 undeploy가 진행된다.

현재 JEUS에서는 성능 향상을 위해 내부적으로 클라이언트에 Home Stub과 EJB Object Stub을 캐시해서 사용하고 있다. 따라서 기존 EJB에 대한 정보를 캐시하고 있는 클라이언트의 경우 Graceful Redeploy 완료 시점 이후에는 기존의 캐시를 삭제하고 새로운 EJB에 대한 정보를 다시 받아와야 한다. EJB 3.x의 경우에는 별도의 설정 없이도 자동으로 변경되지만 EJB 2.x의 경우에는 <use-dynamic-proxy-for-ejb2>가 true로 설정되어 있어야 정상 동작한다(<use-dynamic-proxy-for-ejb2>의 기본값은 true이다).

2.2.6. EAR 애플리케이션 Graceful Redeploy

EAR 애플리케이션의 경우 EAR 안에 속한 WEB, EJB 모듈의 Graceful Redeploy를 보장한다. Standalone 모듈의 Graceful Redeploy와 마찬가지로 기본적으로 웹 모듈의 경우에는 컨텍스트의 세션이 만료될 때까지 이전 서비스를 보장해주고, 이전 Stateless Session Bean의 요청 처리와 Stateful Session Bean의 세션 만료시간까지의 이전 요청 처리를 보장해준다.

단, EAR 애플리케이션은 하나의 애플리케이션이기 때문에 EAR에 속한 하나의 EJB 모듈이 기존 요청 처리를 완료해도 다른 모듈의 이전 요청 처리가 진행 중이면 EJB 모듈을 내릴 수 없다. Graceful Undeploy와 마찬가지로 요청 처리가 끝나지 않은 이전 모듈 안에서 EJB 모듈로 요청할 수 있기 때문에 EAR 애플리케이션의 모든 요청이 정상적으로 처리되는 것을 보장해주려면 하나의 EJB 모듈이나 WEB 모듈만 내리는 것은 불가능하다. EAR 애플리케이션에 속한 모든 웹 모듈이나 EJB 모듈의 요청 처리가 완료되어야 이전 EAR 애플리케이션이 undeploy될 수 있다.



이전 애플리케이션의 application name과 새로운 애플리케이션의 application name은 동일해야 한다. 애플리케이션뿐만 아니라 애플리케이션 안에 속한 각 모듈의 module name 또한 동일해야 한다. application name 설정 방법에 대한 자세한 내용은 Jakarta EE 9 Platform Specification을 참고한다.

3. 애플리케이션

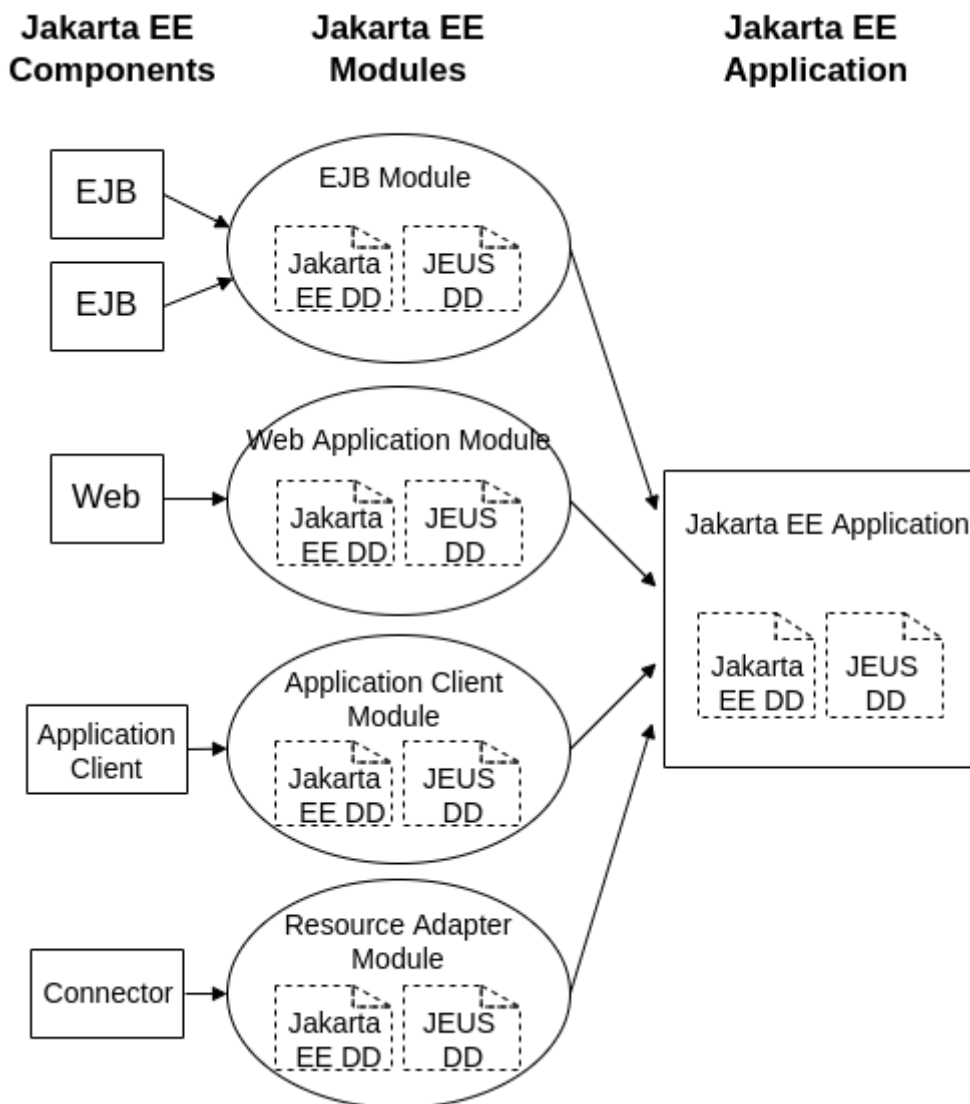
본 장에서는 실제로 모듈과 애플리케이션이 어떻게 구성되어 있으며, 각각의 구성 요소에는 무엇이 있는지에 대해 설명한다. 또한 이렇게 작성된 애플리케이션을 deploy하기 위해 JEUS에서 제공하는 기능에 대해 설명한다.

3.1. 모듈과 애플리케이션

Jakarta EE 애플리케이션은 하나 이상의 모듈로 구성되어 있다. Jakarta EE 모듈은 하나 이상의 동일한 타입인 Jakarta EE 컴포넌트(EJB, 웹 애플리케이션, 애플리케이션 클라이언트, connector)와 Deployment Descriptor(이하 DD)로 구성된다.

DD는 Jakarta EE 표준 DD(application.xml 등)와 JEUS DD(jeus-application-dd.xml 등)가 있으며, 표준 DD에 존재하는 대부분의 설정은 Annotation으로 대체할 수 있다.

다음은 Jakarta EE 모듈 및 애플리케이션 구성이다.



Jakarta EE 모듈 및 애플리케이션 구성

3.1.1. 모듈

Jakarta EE 모듈에는 다음의 4종류가 있다.

- **EJB 모듈 (.jar file)**

EJB(Jakarta™ Enterprise Beans)는 트랜잭션 및 보안 서비스를 이용하는 비즈니스 로직을 구현하기 위한 표준 서버 측 컴포넌트 모델이다. EJB 모듈은 이러한 EJB들을 표현하고 그룹화하기 위한 개념으로 JEUS에서는 JEUS EJB 엔진에 배치할 수 있는 가장 작은 단위를 의미한다. 따라서 1개의 EJB가 배치되어도 반드시 EJB 모듈로 패키지화(.jar file)되어야 한다. EJB 모듈의 자세한 내용은 "JEUS EJB 안내서"를 참고한다.

- **웹(Web) 모듈 (.war file)**

웹 모듈은 클라이언트의 요청에 의한 웹 기반의 서비스(예를 들면, 장바구니에 물품을 추가하거나, 장바구니 안의 물품을 구매하거나 웹 기반의 경매 사이트에서 물건을 사기 위해 브라우징하는 등)를 실행하기 위한 정적 콘텐츠(static content)와 동적 콘텐츠(dynamic content)의 집합이라고 할 수 있다. 웹 애플리케이션 모듈에 대한 자세한 내용은 "JEUS Web Engine 안내서"를 참고한다.

- **애플리케이션 클라이언트(Application Client) 모듈 (.jar file)**

애플리케이션 클라이언트는 별도의 JVM에서 실행되는 클라이언트 프로그램이다. 애플리케이션 클라이언트는 main() 메소드를 호출해서 실행하고, 가상 머신이 종료되면 실행을 마친다.

다른 Jakarta EE 애플리케이션 컴포넌트처럼 애플리케이션 클라이언트는 시스템 서비스를 제공하는 클라이언트 컨테이너(Client Container)에서 동작한다. 클라이언트 컨테이너는 다른 Jakarta EE 컨테이너에 비해서 매우 적은 양의 시스템 리소스를 사용한다. 애플리케이션 클라이언트 모듈에 대한 자세한 내용은 "JEUS Application Client 안내서"를 참고한다.

- **리소스 어댑터(Resource adapter) 모듈 (.rar file)**

리소스 어댑터는 Jakarta EE 커넥터 아키텍처의 중심 컴포넌트로 특정 EIS(Enterprise Information System)용으로 개발되어서 EIS와 상호작용하기 위한 API를 제공한다. 또한 Jakarta EE 애플리케이션 서버와 연동하기 위한 시스템 API도 제공한다. 관리자의 관점에서는 모든 작업이 리소스 어댑터의 설정과 배치만으로 종료된다. 리소스 어댑터 모듈에 대한 자세한 내용은 "JEUS JCA 안내서"를 참고한다.

3.1.2. 애플리케이션

[Jakarta EE 모듈 및 애플리케이션 구성](#)을 보면 Jakarta EE 애플리케이션은 하나 이상의 Jakarta EE 모듈과 2가지 종류의 DD로 구성된다. DD는 Jakarta EE DD(application.xml)와 JEUS DD(jeus-application-dd.xml)로 구성된다. 이때 Jakarta EE 모듈은 1개 이상의 동일한 타입인 컴포넌트들과 DD들로 구성되기 때문에 하나의 컴포넌트만으로도 애플리케이션을 생성할 수 있다.

Jakarta EE 애플리케이션은 확장자가 .ear인 일반적인 JAR Archive 형식의 파일로 다음은 EAR 구성의 간단한 예이다. EAR 파일의 루트 디렉터리에는 애플리케이션에 포함된 각 모듈의 Archive 파일들과 lib, META-INF 디렉터리가 존재한다.

다음은 .ear archive의 구성과 각 각 디렉터리에 대한 설명이다.

```
myApp.ear
```

```

|--lib
|   |--[J]myLib.jar
|-- META-INF
|   |--[X]application.xml
|   |--[X]jeus-application-dd.xml
|--[J]appclient.jar
|--[J]ejb.jar
|--[J]web.war

```

* Legend

- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor

lib

lib 디렉터리는 EAR 애플리케이션의 공통 라이브러리 디렉터리이다. 디렉터리 하위에는 라이브러리 .jar 파일이 올 수 있다. 이렇게 위치한 라이브러리들은 EAR 애플리케이션의 하위 모듈에서 자동으로 클래스 패스에 추가된다. 이 디렉터리는 설정에 의해 다른 디렉터리로 대체될 수 있다.

Java EE 5 버전부터 사용자가 직접 공통 라이브러리 디렉터리를 지정할 수 있다. 지정 시 EAR 애플리케이션 표준 DD인 application.xml에 <library-directory> 태그를 설정하여 지정한다. 만약 application.xml 파일이 없거나 application.xml에 <library-directory> 설정을 하지 않은 경우에는 기본적으로 lib 디렉터리가 사용된다.



JEUS 5부터 지원하는 APP-INF 디렉터리는 공통 라이브러리 디렉터리로 웹 애플리케이션 모듈의 WEB-INF 디렉터리와 비슷한 기능을 제공한다. WEB-INF와 마찬가지로 APP-INF 또한 하위에 classes와 lib 디렉터리를 가지고 있으며, 각각 클래스 파일과 .jar 파일을 포함한다.

APP-INF 아래에 존재하는 classes 디렉터리의 클래스들과 lib 디렉터리의 .jar 파일들은 이 애플리케이션에서 라이브러리로 사용할 수 있다. 하지만 라이브러리 디렉터를 사용하는 것이 표준이므로 JEUS 6 이후에서는 APP-INF 디렉터리보다는 라이브러리 디렉터를 사용할 것을 권장한다.

META-INF

META-INF 디렉터리에는 2개의 DD가 존재한다. Jakarta EE DD인 application.xml과 JEUS DD인 jeus-application-dd.xml이며, 이 파일들은 없어도 무방하다. 이 디렉터리에는 DD 이외에 MANIFEST.MF라는 파일이 존재한다. archive된 EAR 파일의 정보나 클래스 패스를 명시할 수 있다.

3.2. Deployment Descriptor(DD)

Jakarta EE 애플리케이션은 EAR(Enterprise ARchive .ear 파일) 형태로 배포된다.

이 파일에는 EJB 모듈(EJB .jar 파일), 웹 애플리케이션 모듈(.war 파일), 리소스 어댑터 모듈(.rar 파일), 클라이언트

모듈(클라이언트 .jar 파일)과 기타 필요한 Java 클래스를 포함하고 있다. 또한 하나의 모듈로 구성된 Standalone 모듈도 Jakarta EE 애플리케이션의 한 종류이다. 이러한 애플리케이션의 서비스들을 시작하기 위해 JEUS에 모듈 파일을 올리고 제어하는 모든 동작을 **Deployment**라고 한다.

애플리케이션 서버에 배치를 하기 위한 모듈 또는 애플리케이션을 생성하기 위해서는 DD가 필요하다. DD는 Jakarta EE 표준 DD와 JEUS DD가 있다.



표준 DD에서 사용가능한 설정들은 Annotation으로 모두 표현가능해졌기 때문에 EJB와 EAR의 경우 표준 DD와 JEUS DD는 Java EE 5를 만족하는 JEUS 6부터 생략 가능하다. 웹의 경우는 Java EE 6를 만족하는 JEUS 7부터 표준 DD와 JEUS DD를 생략 가능하다.

각 모듈과 애플리케이션에 필요한 DD는 다음과 같다.

	Jakarta EE 표준 DD	JEUS DD
애플리케이션	application.xml	jeus-application-dd.xml
EJB 모듈	ejb-jar.xml	jeus-ejb-dd.xml
웹 애플리케이션 모듈	web.xml	jeus-web-dd.xml
애플리케이션 클라이언트	application-client.xml	jeus-client-dd.xml
리소스 어댑터 모듈	ra.xml	jeus-connector-dd.xml
웹 서비스	webservices.xml	jeus-webservices-dd.xml

각 모듈마다 Jakarta EE 표준 DD 이외에 JEUS를 위한 DD를 별도로 가지고 있는 것처럼 Application descriptor에 대해서도 이에 대응하는 jeus-application-dd.xml이 존재한다. 이 파일은 EAR의 META-INF에 위치한다. 이 파일은 애플리케이션에 대한 부가적인 설정을 하기 위해 사용한다. 만약 이 파일이 EAR이나 Standalone 모듈에 포함되어 있지 않으면 기본 설정으로 deploy가 된다.



Jakarta EE 표준 DD에 대한 보다 자세한 내용은 Jakarta EE 9의 스펙을 참고한다. 각 모듈에 해당하는 JEUS DD 역시 JEUS의 해당 안내서를 참고한다.

jeus-application-dd.xml에는 애플리케이션이 서비스되기 위한 부가적인 정보를 설정할 수 있다. 서버의 deploy에서는 애플리케이션을 deploy할 때 해당 파일에 설정된 정보를 읽어서 애플리케이션을 서비스하기 위한 환경을 구성한다.

다음은 <application> 태그의 설정 항목에 대한 설명이다.

- 보안 설정

애플리케이션에 적용될 보안관련 설정은 다음과 같다. 항목에 대한 자세한 설명은 "JEUS Security 안내서"를 참고한다.

Element	설명
<role-permission>	애플리케이션에서 사용할 principal-role mapping을 지정한다. 이 설정은 애플리케이션 하위의 모든 모듈에 적용된다.

Element	설명
<java-security-permission>	J2SE security manager를 사용할 경우 애플리케이션에게 허용할 permission들을 지정한다.

- 라이브러리 설정

애플리케이션에서 공유 라이브러리를 사용하려고 할 때 아래의 설정을 통해 라이브러리를 지정할 수 있다. 항목에 대한 자세한 설명은 [공유 라이브러리](#)를 참고한다.

Element	설명
<library-ref>	애플리케이션에서 사용할 공유 라이브러리를 지정한다.

- Jakarta EE Name Space 관련 설정

애플리케이션에서 사용할 Name Space에 대한 정보는 application.xml이나 Annotation에 명시할 수 있다. 그에 따른 매핑 정보는 jeus-application-dd.xml에 적을 수 있다. 이 정보들은 Annotation으로 모두 대체 가능하다. Naming 환경에 대한 자세한 내용은 Jakarta EE 9의 스펙을 참고한다. 이 설정에 대한 자세한 설명은 "JEE XML Reference"의 "jeus-application-dd.xml 설정"을 참고한다.

- Classloading 설정

EAR 애플리케이션에 포함된 라이브러리들을 로딩하는 순서를 설정한다. 기본적으로 JAVA의 클래스 로더에 따라 부모 클래스 로더로부터 클래스 로딩을 시도하나 애플리케이션 설정에 의하여 애플리케이션이 가지고 있는 라이브러리를 부모에 우선해 로딩할 필요가 있을 수 있다. 이런 경우 설정을 통해 클래스 로딩의 순서를 변경할 수 있다.

Element	설명
<classloading>	EAR 애플리케이션의 내부 라이브러리를 로딩하는 방식을 설정한다. true로 설정할 경우 EAR 내부에 포함된 라이브러리를 먼저 로딩 후 없을 경우에 부모 클래스 로더의 클래스 로더를 통해 클래스 로딩을 시도한다.

3.3. 애플리케이션에서 라이브러리 사용

본 절에서는 여러 애플리케이션에서 사용할 수 있는 애플리케이션 라이브러리(lib/application)와 공유 라이브러리(Shared Library)를 추가하고, 사용하는 기능에 대해 설명한다. 또한, 새롭게 추가된 라이브러리 배포(Library Deployment) 기능에 대해서도 설명한다.

3.3.1. lib/application 디렉터리

애플리케이션 라이브러리(lib/application)는 애플리케이션 간에 공유되는 라이브러리로 JEUS 시스템에 추가되는 시스템 라이브러리(JEUS_HOME/lib/system)와는 구별된다.

시스템 라이브러리에는 시스템에서 사용하는 라이브러리들이 위치하는 것이고, lib/application에는 애플리케이션이나 기타 사용자가 설정한 클래스(서버 Lifecycle 호출 기능, 사용자 로거, 로거의 필터, 로거의 포맷터

등)에서 사용할 라이브러리를 위한 용도이다.

lib/application에 라이브러리를 위치시키면 라이브러리 파일이 애플리케이션 간에 공유될 수 있다. 따라서 사용자가 항상 애플리케이션에 라이브러리를 함께 패키징(packaging)하지 않아도 된다. 주로 도메인이나 서버에 있는 모든 애플리케이션에서 공통으로 사용될 라이브러리를 여기에 위치시킨다. 라이브러리 형태인 JAR 파일이 올 수도 있다.



lib/application에 같은 라이브러리를 여러 개의 버전으로 설치한 경우 애플리케이션의 의도대로 라이브러리가 선택되지 않을 수 있다. 이 경우 애플리케이션에서 사용할 라이브러리의 버전을 명시할 수 있는 공유 라이브러리를 사용하기 바란다. 공유 라이브러리에 대한 자세한 설명은 [공유 라이브러리](#)를 참고한다.

lib/application은 DOMAIN_HOME과 SERVER_HOME에 위치할 수 있다.

DOMAIN_HOME/lib/application에는 도메인 전체에서 공유할 수 있는 라이브러리를 위치시킨다. 도메인에 설치된 애플리케이션 라이브러리를 다른 머신에 있는 서버에 전달하고 싶다면 다른 머신에서 최초 도메인을 구성할 때 pack-domain 명령어를 통해서 구성하면 DOMAIN_HOME/lib/application도 함께 복사된다. 그 후에 애플리케이션 라이브러리가 추가되거나 변경되는 것은 사용자가 수동으로 동기화해주어야 한다.

특정 서버에서만 사용할 애플리케이션 라이브러리를 설치하고 싶다면 SERVER_HOME/lib/application에 사용할 라이브러리를 위치시킨다. 이는 명령으로 제공하지 않고 있으니 수동으로 위치시켜주어야 한다.



SERVER_HOME/lib/application 디렉터리는 JEUS를 설치할 때 생성되어 있지 않으나 필요하면 직접 생성해서 라이브러리를 위치시킬 수 있다.

클래스 로딩 방식

애플리케이션 라이브러리(lib/application)의 클래스는 JEUS 루트 클래스 로더에 의해서 로딩된다. 따라서 서버에 deploy되는 모든 애플리케이션에서 사용할 수 있다.

시스템 라이브러리(JEUS_HOME/lib/system)와 동일한 클래스 로더에서 로딩되지만, 시스템 라이브러리는 사용자가 접근해서는 안 되는 디렉터리에 존재한다. 하지만 애플리케이션 라이브러리는 사용자를 위해서 사용자가 참조해야 하는 라이브러리를 위치시키는 디렉터리로 두 디렉터리의 역할이 다르다 볼 수 있다.



서버의 클래스 로딩 방식에 대한 자세한 설명은 JEUS Server 안내서의 "클래스 로더의 구조"를 참고한다.

애플리케이션 라이브러리는 SERVER_HOME과 DOMAIN_HOME에 존재한다.

클래스 로더에서 클래스 패스에 추가하는 순서는 SERVER_HOME/lib/application이 먼저이고, DOMAIN_HOME/lib/application이 나중이다. 따라서 SERVER_HOME과 DOMAIN_HOME에 같은 라이브러리가 존재할 경우에는 SERVER_HOME에 설치되어 있는 것이 로딩되기 때문에 DOMAIN_HOME에 존재하는 라이브러리는 무시될 수 있다.

클래스 로더에 라이브러리를 클래스 패스로 추가할 때 라이브러리 파일 이름이 같은 경우에는 서버를 부팅할 때

다음과 같은 WARNING 메시지를 출력한다.

```
Warning [same classpath-name] : [JEUS_HOME/domains/domain1/servers/adminServer
/lib/application/applib.jar] is registered earlier than
[JEUS_HOME/domains/domain1/lib/application/applib.jar] in JEUSClassLoader.
```



위 로그 메시지는 lib/application을 JEUS 루트 클래스 로더에 클래스 패스로 추가할 때 발생한다. 이는 서버를 부팅할 때 클래스 로더를 구성하는 단계에서 진행되는 작업으로 서버 부팅 단계 중 제일 첫 번째 단계이다. 아직 서버 로거가 설정되기 이전의 시점이다. 따라서 이 메시지는 Launcher 로그를 통해서 확인해야 한다.

애플리케이션 라이브러리는 애플리케이션뿐만 아니라 서버에 특정 클래스를 설정해서 사용할 수 있는 부분에서는 모두 사용 가능하다. 그 예로는 서버 Lifecycle 호출 기능과 사용자 로거, 로그 메시지 필터 클래스, 로그 메시지 포맷터 클래스가 있다. 여기에 설정한 클래스에서는 애플리케이션 라이브러리의 라이브러리를 참조하는 것이 아니라 해당 클래스가 lib/application에 위치해야 한다.

3.3.2. 공유 라이브러리

공유 라이브러리(Shared Library)는 애플리케이션 간에 공유되는 라이브러리로 JEUS 시스템 라이브러리(JEUS_HOME/lib/system)와 애플리케이션 라이브러리(DOMAIN_HOME/lib/application, SERVER_HOME/lib/application)와는 구별된다.

공유 라이브러리는 JEUS 전체 시스템에 영향을 주지 않고 각 애플리케이션에서 해당 라이브러리를 사용할 것인지 여부를 설정할 수 있으며, JEUS 재기동 없이 동적으로 추가할 수도 있고, 같은 라이브러리를 여러 개의 버전으로 설치하여 선택적으로 사용할 수 있다.

공유 라이브러리는 다음과 같은 특징을 가진다.

- 라이브러리 파일이 애플리케이션 간에 공유될 수 있고 따라서 사용자가 항상 함께 패키징(packaging)하지 않아도 된다.
- 라이브러리는 서버가 운영 중일 때도 동적으로 추가, 삭제될 수 있다.
- 새로운 버전으로 라이브러리를 추가하고 애플리케이션을 redeploy해서 업그레이드된 라이브러리를 사용할 수 있다.
- 해당 라이브러리의 여러 버전의 구현체를 등록할 수 있고, 어떤 구현체를 사용할지는 Deployment Time에 결정할 수 있다.

각 라이브러리는 2개의 버전(specification, implementation)을 가질 수 있다. 이렇게 함으로써 사용자들은 같은 라이브러리의 여러 개의 버전을 설치할 수 있고, 애플리케이션이 필요한 버전을(highest, minimum or exact) Deployment Time에 동적으로 선택하게 한다.

추후에 여러 버전의 라이브러리를 지원하기 위해서는 처음부터 항상 버전을 명기하는 것을 권장하며, 단순한 use case를 위해서는 버전을 전혀 사용하지 않아도 무방하다. 버전이 명기되지 않았다면 기본적으로 0라는 버전 값으로 내부적으로 해석한다.

라이브러리 설치 및 설정

하나의 라이브러리는 여러 개의 JAR 파일로 구성될 수 있는데 이는 보통 공유 라이브러리 디렉터리인 JEUS_HOME/lib/shared 하위에 위치한다. 그리고 JAR 파일들은 JEUS_HOME/lib/shared/libraries.xml 설정 파일에 다음과 같이 라이브러리로 등록한다.

다음 예에서 'myLibrary'는 2.0 스펙을 구현하는 2.1 버전의 구현체로 정의하여 등록되었다. 이 라이브러리는 여러 개의 JAR 파일(common-logging.jar, common-util.jar 그리고 myLib -2.1 서브 디렉터리에 있는 모든 JAR 파일)로 구성되어 있다.

공유 라이브러리 등록 : <libraries.xml>

```
<library>
  <library-name>myLibrary</library-name>
  <specification-version>2.0</specification-version>
  <implementation-version>2.1</implementation-version>
  <files dir=". ">
    <include name="commons-logging.jar"/>
    <include name="commons-util.jar"/>
  </files>
  <files dir="myLib-2.1"/>
</library>
```

다음은 설정 태그에 대한 설명이다.

- <library-name>, <specification-version>, <implementation-version>
 - 애플리케이션이 해당 라이브러리를 참조할 때 사용한다.
 - <*-version> 필드는 같은 이름의 라이브러리가 여러 버전으로 관리되는 경우에 사용될 수 있다.
- <files>

실제 라이브러리의 클래스 패스를 설정한다. <files> 태그는 다음의 예와 같이 여러 가지 방식으로 클래스 패스를 설정할 수 있다.

공유 라이브러리 <files> 태그

```
<files dir=". ">
  <include name="a.jar"/>
  <include name="b.jar"/>
</files>

<files dir="testa"/>

<files dir="/home/works/lib/testc" />

<files dir="/home/works/lib/testd" mode="classes"/>
```

- dir 값은 JAR 파일이 담긴 디렉터리 또는 classes 디렉터리가 주어질 수 있으며 상대 경로와 절대 경로 모두 사용할 수 있다. 이때 상대 경로의 경우 기반(base) 디렉터리로 공유 라이브러리 디렉터리인 JEUS_HOME/lib/shared에 대한 상대 경로로 해석된다.
- <include> 서브 태그를 이용하여 어떤 JAR 파일들을 포함시킬지 지정할 수 있다. <include> 태그를 설정하지 않으면 해당 디렉터리의 모든 JAR 파일이 포함된다. 이 경우 해당 디렉터리는 deploy 시간에 JAR

파일을 동적으로 검색하게 된다. 따라서, 추후에 설정 변경 없이 JAR 파일을 디렉터리에 추가할 수 있다. JAR 파일 디렉터리가 아니라 클래스 디렉터리를 지정하려면 mode 값을 "classes"로 설정해야 한다.

- 새로운 애플리케이션이 deploy될 때 해당 설정이 수정되었다면 다시 읽어서 처리하기 때문에, 라이브러리는 JEUS가 구동 중에도 동적으로 추가될 수 있다. 따라서, JEUS 재기동 없이도 새로운 라이브러리 혹은 업그레이드된 버전의 라이브러리를 추가할 수 있다.



위에 설명한 설정은 해당 XML을 직접 수정해서 사용해야 한다.

애플리케이션에서 라이브러리 참조

Jakarta EE 애플리케이션이나 Standalone 모듈은 jeus-application-dd.xml, jeus-web-dd.xml 또는 jeus-ejb-dd.xml의 Entry를 통해 등록된 공유 라이브러리를 사용할 수 있다.

다음은 공유 라이브러리 'myLibrary'를 참조하는 예이다.

```
<library-ref>
  <library-name>myLibrary</library-name>
</library-ref>
```

위의 예에서 애플리케이션은 deploy될 때 라이브러리 이름이 'myLibrary'인 라이브러리를 찾고 해당 클래스 패스를 애플리케이션 클래스 패스에 추가한다. 여러 버전의 'myLibrary'가 있는 경우 가장 높은(highest) 버전을 선택하게 된다. 참조되는 라이브러리 버전이 설정되지 않으면, 항상 [Version Ordering Rule](#)에 따라 최상위 버전을 찾게 된다.

다음은 최소 버전이 필요한 경우의 예이다.

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version>
    <value>2.0</value>
  </specification-version>
  <implementation-version>
    <value>2.0</value>
  </implementation-version>
</library-ref>
```

다음은 정확한 버전이 필요한 경우의 예이다.

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version>
    <value>2.0</value>
    <exact-match>true</exact-match>
  </specification-version>
  <implementation-version>
    <value>2.0</value>
    <exact-match>true</exact-match>
  </implementation-version>
</library-ref>
```

다음은 정해진 스펙 버전만 설정하는 경우의 예이다. 이렇게 되면 해당 스펙 버전의 최상의 implementation 버전을 찾게 된다.

```
<library-ref>
  <library-name>myLibrary</library-name>
  <specification-version>
    <value>2.0</value>
    <exact-match>true</exact-match>
  </specification-version>
</library-ref>
```

기본적으로 애플리케이션이 deploy될 때 참조하고 있는 라이브러리를 찾을 수 없다면 WARNING 로그를 보여주지만, deploy는 계속 진행된다. 만약 이런 동작을 원하지 않는다면 다음과 같이 <failon-error> 속성을 설정하여 deploy를 실패시킬 수 있다.

```
<library-ref>
  <library-name>myLibrary</library-name>
  <failon-error>true</failon-error>
</library-ref>
```

클래스 로딩 방식

공유 라이브러리의 클래스는 어느 곳에서 라이브러리 레퍼런스(reference)를 정의하고 있느냐에 따라서 애플리케이션 클래스 로더 혹은 모듈 클래스 로더에 의해서 로딩된다. 예를 들어 'lib1'이 jeus-application-dd.xml에서 레퍼런스로 정의되었다면 EAR 레벨의 애플리케이션 클래스 로더에 의해서 로딩된다. 하지만 jeus-web-dd.xml에 레퍼런스로 정의되었다면 웹 레벨 클래스 로더에 의해서 로딩된다. 각 애플리케이션 클래스 로더에 의해 로딩된 클래스의 경우 해당 애플리케이션에서만 국한(isolated)되며, 이는 라이브러리도 동일하게 적용된다. 따라서 클래스 인스턴스는 애플리케이션 간에 공유되지 않는다.



서버의 클래스 로딩 방식에 대한 자세한 설명은 JEUS Server 안내서의 "클래스 로더의 구조"를 참고한다.

Version Ordering Rule

Version은 "6.2.3-b12"처럼 <fraction_part>(6.2.3)와 <string(non-fraction)_part>(-b12)를 가질 수 있다.

```
Version ::= <fraction_part> | <string_part> | <fraction_part> <string_part>
fraction_part ::= <integer> | <integer> "." <fraction_part>
string_part ::= <non-numeric> <character>*
```

Version을 ordering하는 규칙은 다음과 같다.

- <fraction_part>를 먼저 수치적으로 비교한다. major, minor 순서로 비교된다.
- <fraction_part>가 동일하면 <string_part>를 비교한다. 이때 비교는 string 비교 방식을 따른다.

다음은 ordering 규칙에 따른 순서의 예이다.

6.0 < 6.2.3 < 6.2.3-b12 < 6.2.3-beta < 6.2.4

3.3.3. Library Deployment

JEUS 8부터는 라이브러리를 Master Server를 통해 관리할 수 있도록 라이브러리 배포(Library Deployment) 기능을 제공한다. 라이브러리 배포 기능을 사용하면 사용자는 참조할 라이브러리를 JEUS가 제공하는 툴을 통해 도메인에 배포한 후 설정을 통해 애플리케이션이 이를 참조하도록 할 수 있다. 이를 통해 서로 다른 애플리케이션이 서로 다른 버전의 동일한 라이브러리를 참조할 때 발생할 수 있는 클래스 충돌 문제 등을 최소화 할 수 있다.

라이브러리 배포 기능을 사용하여 얻을 수 있는 장점은 아래와 같다.

- 애플리케이션마다 사용하는 라이브러리를 같이 패키징할 필요가 없다. 이는 같은 라이브러리를 여러 애플리케이션에서 사용하고자 할 때, 각각 패키징을 할 경우 발생하는 편의성 문제나 중복 클래스 로딩으로 인한 리소스 사용 문제를 해결할 수 있다.
- 애플리케이션 배포할 때 사용할 라이브러리를 지정할 수 있다. 동일한 라이브러리가 복수 배포되어 있는 경우 버전을 지정함으로써 사용할 라이브러리를 지정할 수 있다.

주요 기능

애플리케이션에서 라이브러리를 사용하기 위해서는 설치 및 배포 과정을 거쳐야 한다. 라이브러리 배포를 위해 제공하는 주요 기능들은 다음과 같다.

1. 라이브러리 설치(Install)

라이브러리를 배포하기 위해서는 먼저 Master Server가 있는 장비의 도메인 디렉터리에 해당 라이브러리를 설치해야 한다. 설치하는 콘솔 명령어를 통해 이루어지며, 설치 작업에 필요한 정보들은 다음과 같다.

구분	설명
라이브러리 식별자(ID)	배포, 삭제 및 참조할 때 사용할 라이브러리에 대한 이름이다. 도메인 내에서 고유한 값이어야 하며, 지정하지 않은 경우 설치가 되지 않는다.
라이브러리 버전	설치할 라이브러리에 대한 버전을 지정할 수 있다. 지정하지 않은 경우에는 1.0으로 간주한다.
라이브러리 경로	설치할 라이브러리가 위치한 경로를 지정한다. 지정하지 않은 경우 설치가 되지 않는다.

설치 작업이 완료되면 라이브러리 파일(들)은 DOMAIN_HOME/.libraries/LIBRARY_ID/VERSION 아래에 위치한다.



설치한 라이브러리에 대한 정보는 설정에 별도로 기록하지 않는다. Master Server가 라이브러리 설치 상태를 파악할 필요가 있을 경우에는 DOMAIN_HOME/.libraries 아래에 디렉터리 구조를 탐색한다. 따라서 임의로 해당 디렉터리를 변경하는 것은 권장하지 않는다.

2. 라이브러리 배포(deployment)

설치한 라이브러리는 콘솔 명령어를 사용하여 배포할 수 있다. 배포할 때 해당 라이브러리를 사용할 서버나 클러스터를 명시하거나, 모든 서버를 대상으로 하도록 명시할 수 있다. 배포가 성공적으로 이루어지면 Master Server는 해당 내용을 설정에 기록하여 추후 재기동 할 때에도 라이브러리 설치 상태를 유지할 수 있도록 한다.

3. 라이브러리 비활성화(undeployment)

라이브러리를 더 이상 사용하지 않을 경우 콘솔 명령어를 통해 해당 라이브러리를 비활성화할 수 있다. 작업이 완료되면 Master Server는 비활성화한 라이브러리에 대한 정보를 설정에서 제거하나 라이브러리 파일 자체는 삭제하지 않고 유지한다.

4. 라이브러리 삭제(Uninstall)

콘솔 명령어를 통해 라이브러리 파일을 삭제할 수 있다.

3.3.3.1. 라이브러리 설치 및 삭제

본 절에서는 라이브러리를 설치하고 삭제하는 방법을 설명한다.

콘솔 툴 사용

다음은 콘솔 툴(jeusadmin)을 사용하여 라이브러리를 설치 및 삭제하는 방법에 대해 설명이다.

• 현재 설치되어 있는 라이브러리에 대한 정보 확인

library-info 명령어를 실행하면 현재 설치 및 배포되어 있는 라이브러리에 대한 목록을 확인할 수 있다.

```
[MASTER]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
(No data available)
=====
```

• 라이브러리 설치

install-library 명령어를 실행하면 라이브러리를 설치할 수 있다. 라이브러리 식별자(ID) 및 경로를 지정하지 않는 경우 설치가 되지 않으므로 반드시 입력해야 한다.

```
[MASTER]domain1.adminServer>install-library log4j -path /usr/lib/apache-log4j-1.2.17/log4j-1.2.17.jar -version 1.2.17
Successfully installed the library [log4j] version [1.2.17].
[MASTER]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target | Target Clusters | Applicatio |
+-----+-----+-----+-----+-----+-----+
=====
```

			Servers		ns
log4j	1.2.17	INSTALLED			

• 라이브러리 배포

라이브러리를 배포하기 위해서는 **deploy-library** 명령어를 사용한다. 배포할 때 라이브러리를 사용할 대상 (서버, 클러스터 또는 모든 서버)를 지정해야 한다.

```
[MASTER]domain1.adminServer>deploy-library log4j -servers adminServer
deploy the library [log4j] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
| log4j     | 1.2.17 | RUNNING| adminServer   |                  |              |
+-----+-----+-----+-----+-----+-----+
=====
```

• 라이브러리 비활성화

배포한 라이브러리를 더 이상 사용하지 않게 된 경우 **undeploy-library** 명령어를 통해 해당 라이브러리를 비활성화할 수 있다. 사용하지 않는 라이브러리를 비활성화하지 않아도 서버의 기능에는 문제가 없으나, 서버를 기동할 때 라이브러리 동기화 등 불필요한 작업이 발생할 수 있다.

```
[MASTER]domain1.adminServer>undeploy-library log4j
undeploy the library [log4j] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applicatio |
|           |        |      | Servers      |                  | ns         |
+-----+-----+-----+-----+-----+-----+
| log4j     | 1.2.17 | INSTALLED|              |                  |            |
+-----+-----+-----+-----+-----+-----+
=====
```

• 라이브러리 삭제

라이브러리를 도메인에서 완전히 제거하고자 하는 경우 **uninstall-library** 명령어를 사용한다. 이미 배포가 되어 있는 라이브러리는 삭제할 수 없으며, 삭제를 하기 위해서는 비활성화 과정을 거쳐야 한다.

```
[MASTER]domain1.adminServer>uninstall-library log4j
uninstall the library [log4j] succeeded. : Successfully deleted [log4j].
[MASTER]domain1.adminServer>library-info
Library information
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
(No data available)
=====
```

3.3.3.2. 애플리케이션에서 라이브러리 참조

애플리케이션을 배포할 때 참조할 라이브러리를 지정할 수 있다. 애플리케이션을 deploy할 때 참조할 라이브러리의 식별자와 버전을 지정할 수 있다. 버전을 지정하지 않은 경우에는 가장 높은(최신) 버전의 라이브러리를 지정한 것으로 간주한다.

JEUS는 배포할 때 지정한 정보를 사용하여 해당 라이브러리를 애플리케이션이 참조할 수 있도록 연결하는 작업을 수행한다. 애플리케이션이 참조하는 라이브러리에 대한 정보는 다른 배포 정보와 같이 domain.xml에 기록된다.

콘솔 툴 사용

deploy-application 명령어를 수행할 때 참조할 라이브러리의 식별자와 버전을 지정한다.

```
[MASTER]domain1.adminServer>deploy-application sample -lib log4j -version 1.2.17 -servers adminServer
deploy the application for the application [sample] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
=====
+-----+-----+-----+-----+-----+-----+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----+-----+-----+-----+-----+-----+
| log4j      | 1.2.17 | RUNNING| adminServer   |                  | sample      |
+-----+-----+-----+-----+-----+-----+
=====
```

4. 애플리케이션 작성 및 Deploy

본 장에서는 Jakarta EE 애플리케이션 파일(EAR)을 작성하고 이를 JEUS에 deploy하는 방법에 대해 설명한다. 또한 JEUS에서 제공하는 애플리케이션을 deploy할 수 있는 툴을 사용하여 JEUS 서버에 애플리케이션을 deploy하는 방법과 그 외 관련된 작업에 대해 설명한다.

4.1. 애플리케이션 작성

본 절에서는 작성된 각각의 모듈을 포함하는 애플리케이션 작성 방법 중 jar 유틸리티를 사용해서 Jakarta EE 애플리케이션을 직접 작성하는 방법에 대해서만 설명한다.

EAR 파일을 작성하기 전에 우선 포함될 모듈을 작성해야 한다. EAR 파일에 포함되는 모듈은 EJB 모듈인 JAR 파일과 웹 애플리케이션 모듈인 WAR 파일, 리소스 어댑터 모듈인 RAR 파일 등이 있다. 모듈 파일을 작성하는 자세한 내용에 대해서는 각 해당 안내서에서 다루고 있으므로 해당 안내서를 참고한다.

다음은 애플리케이션 작성 환경에 대한 설명이다.

- myApp.ear이라는 EAR 파일을 작성한다. 이 파일은 ejb.jar라는 EJB 모듈과 web.war라는 웹 애플리케이션 모듈, appclient.jar 애플리케이션 클라이언트 모듈을 포함하고 있다.
- 애플리케이션 myApp.ear을 먼저 도메인의 Master Server인 "adminServer"로 install하고, 서버 "server1"에 deploy한다.

다음의 순서로 애플리케이션을 작성한다.

1. EAR 파일에 포함될 모듈을 작성한다.
2. JAR, WAR, RAR 파일과 같은 디렉터리에서 META-INF 디렉터리를 생성한다.
3. application.xml 파일을 생성(EAR 파일의 모듈을 포함시킨다)해서 META-INF 디렉터리에 복사한다.

<application.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<application version="9"
  xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
    https://jakarta.ee/xml/ns/jakartaee/application_9.xsd">
  <description>Application description</description>
  <display-name>Sample application</display-name>
  <module>
    <ejb>ejb.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>web.war</web-uri>
      <context-root>hello</context-root>
    </web>
  </module>
  <module>
    <java>appclient.jar</java>
  </module>
```



```
</application>
```

4. 다음과 같이 jar 유틸리티를 사용해서 명령어를 실행하면 myApp.ear 파일이 생성된다.

```
> jar cf myApp.ear ejb.jar web.war appclient.jar META-INF
```



META-INF 디렉터리가 대문자인 것에 주의한다. 소문자로 작성할 경우에는 문제가 발생한다.

4.2. Deploy 명령어

JEUS가 제공하는 Deploy 관련 툴에서는 다음과 같은 작업들을 지원한다.

툴	설명
distribute	애플리케이션 파일을 대상 서버나 클러스터로 복사하고, 애플리케이션을 서비스하기 위한 사전 작업을 한다. 대상으로 설정한 서버나 클러스터 중 하나라도 distribute에 실패하면 전체 작업을 실패로 간주하고 distribute에 성공한 서버에서 모두 undeploy한다.
deploy	애플리케이션 파일을 대상 서버로 복사하고, 애플리케이션을 서비스하기 위한 사전 작업을 한다. 이 작업이 성공하면 애플리케이션을 실행시킨다. 대상으로 설정한 서버나 클러스터 중 하나라도 distribute에 실패하면 전체 작업을 실패로 간주하고 distribute에 성공한 서버에서 모두 undeploy한다. Start 작업에서 실패한 서버가 있더라도 하나라도 서비스 가능한 상태가 있으면 애플리케이션 Start 작업은 성공이다.
start	대상 서버에 distribute(배포)되어 있는 애플리케이션을 동작시키는 작업이다. Start 작업에서 실패한 서버가 있더라도 하나라도 서비스 가능한 상태가 있으면 애플리케이션 Start 작업은 성공이다.
stop	대상 서버에 배치되어 실행 중인 애플리케이션을 일시정지시키는 작업이다. 이때 애플리케이션을 대상 서버에서 삭제하지 않는다. 또한 애플리케이션 이름으로 start하거나 redeploy할 수도 있다.
undeploy	배치되어 실행 중인 애플리케이션을 정지시키고, deploy되어 있는 대상 서버나 클러스터에서 애플리케이션을 제거하는 작업이다.
redeploy	deploy되어 실행 중인 애플리케이션의 내용이 변경된 경우에 그 내용을 현재 애플리케이션에 반영하여 다시 deploy하는 작업이다. 모든 애플리케이션을 재배포할 때 Deploy 작업이 하나라도 실패하면 모든 애플리케이션은 정지된다.

다음 작업들은 JEUS에서만 제공하는 Deploy 작업이다. 이 작업은 JEUS가 제공하는 툴을 사용하여 진행할 수 있다.

작업	설명
Add Application Target	<p>deploy나 distribute된 애플리케이션에 대상으로 특정 서버 또는 클러스터를 추가한다. 이미 서비스되고 있는 애플리케이션을 서비스하는 대상을 확장하려고 할 때 사용하는 작업이다.</p> <p>대상은 여러 서버 또는 여러 클러스터를 설정할 수 있고, 이 작업이 대상으로 설정한 서버나 클러스터에서 실패한 경우 전체 작업 실패로 간주하고 기존에 서비스되고 있던 서버를 제외한 새로 추가하려는 서버에서는 모두 undeploy된다.</p>
Remove Application Target	<p>deploy나 distribute된 애플리케이션에서 대상으로 설정된 특정 서버 또는 클러스터를 삭제한다. 이미 서비스되고 있는 애플리케이션을 서비스하는 대상을 축소하려고 할 때 사용하는 작업이다.</p> <p>대상은 여러 서버 또는 여러 클러스터를 설정할 수 있다.</p>

4.3. 애플리케이션 제어 및 모니터링

본 절에서는 애플리케이션 제어 및 모니터링 방법에 대해서 설명한다.

4.3.1. 애플리케이션을 도메인에 Install

다음은 애플리케이션을 도메인에 install하는 방법에 대한 설명이다.

콘솔 툴 사용

install-application 명령어를 통해 애플리케이션을 도메인에 install하여 `INSTALL_HOME(DOMAIN_HOME/.applications)`에 위치시킨다. 이때 설치할 애플리케이션 파일 경로를 지정해야 하며 '-id' 옵션을 사용하여 애플리케이션의 ID를 지정할 수 있다.

만약 ID를 지정하지 않을 경우 JEUS가 자동으로 애플리케이션 ID를 지정한다. 예를 들어 애플리케이션 파일 이름이 `myApp.ear`인 경우 ID를 지정하지 않은 경우는 `myApp_ear`이 애플리케이션의 ID가 된다.

```
-----
using install-application command for application install
-----

[MASTER]domain1.adminServer>install-application -id myApp /usr/apphome/myApp.ear
Successfully installed the application [myApp].

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati|Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    |           | INSTAL|         |         | ${INSTALL_HOME}/myApp/m|
|          |           | LED   |         |         | yApp.ear             |
+-----+-----+-----+-----+-----+-----+
```

=====

uninstall-application 명령어에 대한 자세한 내용은 JEUS Reference 안내서의 "uninstall-application"을 참고한다.

4.3.2. 도메인에서 애플리케이션 Uninstall

애플리케이션이 도메인에 더 이상 사용되지 않는다면 도메인에서 삭제할 수 있다. 애플리케이션이 INSTALLED 상태이거나 DEPLOYED 상태인 경우에 애플리케이션에 Uninstall 작업을 할 수 있다.

다음은 애플리케이션을 도메인에서 Uninstall하는 방법에 대한 설명이다.

콘솔 툴 사용

uninstall-application 명령어를 통해 애플리케이션을 uninstall할 수 있다. 명령어 실행 시 uninstall할 애플리케이션 ID를 명시해야 한다.

```
[MASTER]domain1.adminServer>uninstall-application myApp
uninstall the application for the application [myApp] succeeded. : Successfully deleted [myApp].
```

```
[MASTER]domain1.adminServer>application-info
No applications exist in this domain.
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Application | Application | State | Server | Cluster | Application |
| ID          | Type       |      | Targets| Targets | Path        |
+-----+-----+-----+-----+-----+-----+
(No data available)
=====
```

4.3.3. 애플리케이션 Deploy

install한 애플리케이션은 deploy 명령어를 통해 deploy하여 서비스 가능한 상태로 만들 수 있다. 본 절에서는 애플리케이션을 deploy하는 3가지 방법에 대해 설명한다.

- 도메인에 install한 애플리케이션 Deploy
- 애플리케이션 저장소에 있는 애플리케이션 Deploy
- path를 지정하여 Deploy

콘솔 툴을 통한 Run-time Deploy 방법은 다음의 사항을 가정하여 설명한다.

- 애플리케이션은 [애플리케이션 작성](#)에서 작성한 myApp.ear을 사용한다.
- JEUS의 도메인 이름은 'domain1'이고, 애플리케이션을 deploy할 서버는 'server1'이다.



deploy-application 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS

Reference 안내서의 "deploy-application"을 참고한다.

JEUS에서는 콘솔 툴에서 애플리케이션을 Run-time Deploy할 경우 애플리케이션의 위치에 따라 다음의 3가지 방법을 제공한다. 각 방법은 deploy, stop, start, redeploy, undeploy의 순서대로 deploy 명령어를 실행하며, 각 명령어를 실행한 뒤에는 **applist**를 수행하여 각 단계에서의 애플리케이션 상태를 확인한다.

JEUS를 직접 관리할 수 있는 콘솔 툴을 통해 Master Server에 연결하여 서버나 클러스터의 제어 및 모니터링을 할 수 있으며, 애플리케이션을 deploy할 수도 있다.

다음은 콘솔 툴에서 사용하는 애플리케이션의 Deploy 관련 명령어이다.

명령어	설명
distribute-application	애플리케이션을 deploy할 수 있도록 배치 대상에 애플리케이션을 복사하고, 애플리케이션 서비스를 위한 사전 준비작업을 한다.
deploy-application	애플리케이션을 deploy한다. Deploy란 Deploy 대상에 애플리케이션을 복사하고(Distribute), 애플리케이션의 서비스를 시작(Start)하는 것이다. Deploy가 성공적으로 완료되면 RUNNING 상태가 되며, Distribute 완료 후 서비스 단계에서 장애가 발생하면 DISTRIBUTED 상태에 머물러 있다.
start-application	DISTRIBUTED 상태에 있는 애플리케이션을 실행한다. 작업이 수행되는 동안에는 STARTING 상태로 성공적으로 완료되면 RUNNING 상태가 된다.
stop-application	RUNNING 상태에 있는 애플리케이션을 정지한다. 작업이 수행되는 동안에는 STOPPING 상태로 성공적으로 완료되면 DISTRIBUTED 상태가 된다.
redeploy-application	Deploy가 완료된 애플리케이션에 변경된 내용이 있는 경우 변경 내용을 반영하여 다시 deploy한다. Deploy와 같은 단계로 이루어지며 각 단계에서의 상태 또한 같다.
undeploy-application	애플리케이션 서비스를 종료하고 배치 대상에서 애플리케이션을 제거한다.
application-info	도메인에 존재하는 애플리케이션의 정보를 출력한다.
add-application-target	Deploy되어 있는 애플리케이션에 서비스 대상을 추가한다. 서비스 대상은 서버나 클러스터가 될 수 있다.
remove-application-target	Deploy되어 있는 애플리케이션에서 서비스 대상을 삭제한다. 서비스 대상은 서버나 클러스터가 될 수 있다.



콘솔 툴에 대한 설명 및 각 명령어에 대한 자세한 내용은 JEUS Reference 안내서의 "jeusadmin"을 참고한다.

4.3.3.1. 도메인에 install한 애플리케이션 Deploy

본 절에서는 도메인에 install한 애플리케이션을 Deploy하는 방법에 대해 설명한다.

콘솔 툴 사용

install-application 명령어를 통해 애플리케이션을 도메인에 install하여

INSTALL_HOME(DOMAIN_HOME/.applications)에 위치시킨다. 이때 애플리케이션의 ID를 옵션으로 줄 수 있다.

만약 ID를 주지 않은 경우 애플리케이션 파일 이름이 myApp.ear이므로 myApp_ear이 애플리케이션의 ID가 된다.

애플리케이션을 도메인에 install한 후 **deploy-application** 명령어를 사용해 대상 서버에 애플리케이션을 deploy한다.

```
-----  
using deploy command for application with install application  
-----
```

```
[MASTER]domain1.adminServer>install-application -id myApp /usr/apphome/myApp.ear  
Successfully installed the application [myApp].
```

```
[MASTER]domain1.adminServer>application-info  
Application information for the domain [domain1].
```

```
=====
```

Applicati on ID	Application Type	State	Server Targets	Cluster Targets	Application Path
myApp		INSTAL LED			\${INSTALL_HOME}/myApp/m yApp.ear

```
=====
```

```
[MASTER]domain1.adminServer>deploy myApp -servers server1  
deploy the application for the application [myApp] succeeded.
```

```
[MASTER]domain1.adminServer>application-info  
Application information for the domain [domain1].
```

```
=====
```

Applicati on ID	Application Type	State	Server Targets	Cluster Targets	Application Path
myApp	EAR	RUNNING	server1		\${INSTALL_HOME}/myAp p/myApp.ear

```
=====
```

```
[MASTER]domain1.adminServer>stop-application myApp  
stop the application for the application [myApp] succeeded.
```

```
[MASTER]domain1.adminServer>application-info  
Application information for the domain [domain1].
```

```
=====
```

Applicati on ID	Application Type	State	Server Targets	Cluster Targets	Application Path
myApp	EAR	DISTRIB UTED	server1		\${INSTALL_HOME}/myApp/m yApp.ear

```
=====
```

```
[MASTER]domain1.adminServer>start-application myApp
start the application for the application [myApp] succeeded.
```

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR       | RUNNING | server1 |          | ${INSTALL_HOME}/myAp|
|          |           |         |         |          | p/myApp.ear         |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[MASTER]domain1.adminServer>redploy-application myApp
redploy application on JEUS Master Server for the application [myApp] succeeded.
```

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR       | RUNNING | server1 |          | ${INSTALL_HOME}/myAp|
|          |           |         |         |          | p/myApp.ear         |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[MASTER]domain1.adminServer>undeploy myApp
Undeploying [myApp] (This may take time due to graceful undeployment) .....
undeploy the application for the application [myApp] succeeded.
successfully undeployed (elapsed = 415ms)
```

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR       | INSTAL|         |          | ${INSTALL_HOME}/myApp/m|
|          |           | LED   |         |          | yApp.ear            |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[MASTER]domain1.adminServer>uninstall-application myApp
uninstall the application for the application [myApp] succeeded. : Successfully deleted [myApp].
```

```
[MASTER]domain1.adminServer>application-info
No applications exist in this domain.
Application information for the domain [domain1].
```

```
=====
+-----+-----+-----+-----+-----+-----+
| Application| Application | State | Server | Cluster | Application |
| ID         | Type       |      | Targets| Targets | Path        |
+-----+-----+-----+-----+-----+-----+
=====
```

(No data available)

=====

4.3.3.2. 애플리케이션 저장소에 있는 애플리케이션 Deploy

애플리케이션 저장소를 추가하고 나면 저장소에 존재하는 애플리케이션을 deploy한다. 애플리케이션 저장소의 기능과 저장소를 추가, 삭제하는 방법에 대한 자세한 내용은 [애플리케이션 저장소 추가/삭제/조회](#)를 참고한다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 애플리케이션 저장소에 있는 애플리케이션 deploy하는 예제이다.

```
[MASTER]domain1.adminServer>add-application-repository /home/user1/apps
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or list-application-repositories"
```

```
[MASTER]domain1.suok>list-application-repositories
```

Application Repositories

```
=====
+-----+-----+-----+-----+
| Path to Application Repository |
+-----+-----+-----+-----+
| /home/user1/apps |
+-----+-----+-----+-----+
=====
```

```
[MASTER]domain1.adminServer>application-info
```

Application information for the domain [domain1].

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                  |
+-----+-----+-----+-----+-----+-----+
| exploded |           | INSTAL|         |         | /home/user1/apps/explod|
|          |           | LED   |         |         | ed                    |
+-----+-----+-----+-----+-----+-----+
=====
```

```
[MASTER]domain1.adminServer>deploy exploded -servers server1
```

deploy the application for the application [exploded.war] succeeded.

```
[MASTER]domain1.adminServer>application-info
```

Application information for the domain [domain1].

```
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                  |
+-----+-----+-----+-----+-----+-----+
| exploded | WAR       | RUNNING | server1 |         | /home/user1/apps/exp|
|          |           |         |         |         | loded                |
+-----+-----+-----+-----+-----+-----+
=====
```

4.3.3.3. path를 지정하여 Deploy

다음은 상위 디렉터리를 애플리케이션 저장소로 추가할 수 없는 경우에 path 옵션을 설정하고 deploy하는 방법에 대한 설명이다. Master Server가 존재하는 머신에 있는 애플리케이션의 절대 경로를 설정하고 deploy하는 기능에 대한 자세한 내용은 [path를 지정하여 Deploy](#)를 참고한다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 path를 지정하여 deploy하는 방법이다.

```
[MASTER]domain1.adminServer>deploy -path /home/user1/apps/myApp.ear -servers server1
deploy the application for the application [/home/user1/apps/myApp.ear] succeeded.
```

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

=====					
Applicati on ID	Application Type	State	Target Servers	Target Clusters	Application Path
myApp.ear	EAR	RUNNING	server1		/home/user1/apps/ myApp.ear
=====					

4.3.4. 애플리케이션 Redeploy

애플리케이션을 수정한 경우 redeploy 명령어를 사용하여 애플리케이션을 다시 deploy하고, 수정 사항을 반영할 수 있다.

콘솔 툴 사용

다음은 콘솔 툴을 사용하여 애플리케이션을 redeploy하는 과정에 대한 설명이다.

1. 도메인에 install한 애플리케이션의 경우 수정한 애플리케이션을 새로 설치한다. 기존에 설치한 애플리케이션을 교체하는 것이기 때문에 install-application 수행 시 -f 옵션을 사용해야 한다.

path를 지정하여 deploy한 경우에는 지정한 path에 있던 기존 애플리케이션 파일을 수정한 애플리케이션 파일로 교체한다.

```
[MASTER]domain1.adminServer>install-application -id myApp -f /home/user1/apps/myApp.ear
Successfully installed the application [myApp].
```

2. **redeploy-application** 명령어를 통해 애플리케이션을 redeploy한다. 명령어 실행 시 redeploy할 애플리케이션 ID를 반드시 입력해야 한다.

```
[MASTER]domain1.adminServer>redeploy-application myApp
```


redploy application on JEUS Master Server for the application [myApp] succeeded.



redploy-application 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "redploy-application"을 참고한다.

4.3.5. 애플리케이션 Undeploy

서비스 중인 애플리케이션을 중단하고 undeploy하는 작업은 undeploy 명령어를 통해 가능하다. undeploy를 수행하면 도메인에서 해당 애플리케이션이 삭제된다.

콘솔 툴 사용

다음은 콘솔 툴을 사용하여 애플리케이션을 undeploy하는 방법에 대한 설명이다.

undeploy-application 명령어를 사용하여 애플리케이션을 undeploy할 수 있다. undeploy-application 명령어 입력 시 undeploy할 애플리케이션 ID를 반드시 입력해야 한다.

```
[MASTER]domain1.adminServer>undeploy-application myApp
Undeploying [myApp] (This may take time due to graceful undeployment) .....
undeploy the application for the application [myApp] succeeded.
successfully undeployed (elapsed = 82ms)
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Applicati | State | Target | Target | Application Path |
| ion ID   | on Type   |       | Servers | Clusters |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR       | INSTA |         |         | ${INSTALL_HOME}/myApp/ |
|          |           | LLED  |         |         | myApp.ear              |
+-----+-----+-----+-----+-----+-----+
=====
[MASTER]domain1.adminServer>
```



- Install 과정을 거치지 않고 deploy한 애플리케이션을 undeploy하는 경우에는 INSTALLED 상태로 표시되지 않고 목록에서 바로 삭제된다.
- **undeploy-application** 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "undeploy-application"을 참고한다.

4.3.6. 애플리케이션 시작

애플리케이션을 deploy할 때 **'Only Distribute'** 옵션을 주고 distribute만 수행했거나, 서비스 중인 애플리케이션을 stop 명령어를 수행하여 DISTRIBUTED 상태인 애플리케이션을 다시 서비스하도록 하려면 Start 작업을 수행해야 한다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 중지된 애플리케이션이 서비스를 시작하는 방법에 대한 설명이다.

start-application 명령어를 수행하여 애플리케이션을 start할 수 있다. start-application 명령어 실행 시 start할 애플리케이션 ID를 반드시 입력해야 한다.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Applicati | State | Target | Target | Application Path |
| ion ID   | on Type   |       | Servers | Clusters |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR       | DISTRI | server1 |          | ${INSTALL_HOME}/myApp/ |
|          |           | BUTED  |         |          | myApp.ear              |
+-----+-----+-----+-----+-----+-----+
=====

[MASTER]domain1.adminServer>start-application myApp
start the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Application | State | Target | Target | Application Path |
| ion ID   | Type        |       | Servers | Clusters |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR         | RUNNING | server1 |          | ${INSTALL_HOME}/myA |
|          |             |         |         |          | pp/myApp.ear        |
+-----+-----+-----+-----+-----+-----+
=====

[MASTER]domain1.adminServer>
```



start-application 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "start-application"을 참고한다.

4.3.7. 애플리케이션 중지

애플리케이션을 undeploy하지 않고, 잠시 서비스만 중단하고자 하는 경우 다음과 같은 방법을 사용하여 애플리케이션을 stop할 수 있다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 애플리케이션이 서비스를 중지하는 방법에 대한 설명이다.

stop-application 명령어를 사용하여 애플리케이션 서비스를 중지할 수 있다. stop-application 명령어 실행 시 중단할 애플리케이션 ID를 반드시 입력해야 한다.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```

=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Application | State | Target | Target | Application Path |
| ion ID   | Type       |      | Servers| Clusters|                  |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1 |          | ${INSTALL_HOME}/myA |
|          |            |        |         |          | pp/myApp.ear         |
+-----+-----+-----+-----+-----+-----+

[MASTER]domain1.adminServer>stop-application myApp
stop the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Applicati | State | Target | Target | Application Path |
| ion ID   | on Type   |      | Servers| Clusters|                  |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | DISTRI | server1 |          | ${INSTALL_HOME}/myApp/ |
|          |            | BUTED  |         |          | myApp.ear             |
+-----+-----+-----+-----+-----+-----+

[MASTER]domain1.adminServer>

```



stop-application 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "stop-application"을 참고한다.

4.3.8. 서비스 중인 애플리케이션에 서버 추가

애플리케이션 서비스를 확장하기 위해 현재 실행 중인 애플리케이션의 대상에 서버나 클러스터를 추가할 수 있다. 이미 도메인에 존재하는 애플리케이션을 특정 서버에만 deploy할 수는 없기 때문에 서비스 대상을 추가(add-target)하는 별도의 명령어를 지원한다. 애플리케이션에 서비스 대상을 추가하는 명령(add-target)은 애플리케이션이 RUNNING 상태이거나 DISTRIBUTED 상태일 때 가능하다.



애플리케이션 서비스 대상에 서버를 추가하려면 서버가 하나 더 존재해야 한다. 서버를 추가하는 방법은 JEUS Server 안내서의 "서버 추가"를 참고한다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 서비스 중인 애플리케이션에 서버를 추가하는 방법에 대한 설명이다.

add-application-target 명령어를 사용하여 실행 중인 애플리케이션의 대상을 추가할 수 있다. add-application-target 명령어 실행 시 대상을 추가할 애플리케이션 ID와 애플리케이션을 추가할 서버(또는 클러스터) 목록을 반드시 지정해야 한다.

```

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====

```

```

+-----+-----+-----+-----+-----+-----+
| Applicat | Application | State | Target | Target | Application Path |
| ion ID   | Type       |      | Servers| Clusters|                  |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1 |          | ${INSTALL_HOME}/myA |
|          |            |        |         |          | pp/myApp.ear         |
+-----+-----+-----+-----+-----+-----+

=====
[MASTER]domain1.adminServer>add-application-target myApp -servers server2
add a target server or cluster to the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Applicati | State | Target | Target | Application Path |
| ion ID   | on Type   |      | Servers| Clusters|                  |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1,s |          | ${INSTALL_HOME}/myA |
|          |            |        | erver2   |          | pp/myApp.ear         |
+-----+-----+-----+-----+-----+-----+

=====
[MASTER]domain1.adminServer>

```



add-application-target 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "add-application-target"을 참고한다.

4.3.9. 서비스 중인 애플리케이션에서 서비스 중인 서버 삭제

애플리케이션 서비스를 축소하기 위해 애플리케이션을 서비스하고 있는 대상에서 서버나 클러스터를 삭제하는 기능을 제공한다. 이미 도메인에 존재하는 애플리케이션을 특정 서버에만 undeploy할 수 없기 때문에 서비스 대상을 삭제(remove-target)하는 별도의 명령어를 지원한다. 애플리케이션에 서비스 대상을 삭제하는 명령은 애플리케이션이 RUNNING 상태 또는 DISTRIBUTED 상태일 때 가능하다.

콘솔 툴 사용

다음은 콘솔 툴을 사용해서 서비스 중인 애플리케이션에서 서비스 중인 서버를 삭제하는 방법에 대한 설명이다.

remove-application-target 명령어를 사용하여 실행 중인 애플리케이션의 대상을 삭제할 수 있다. remove-application-target 명령어 실행 시 대상을 삭제할 애플리케이션 ID와 삭제할 대상 서버(또는 클러스터) 목록을 반드시 입력해야 한다.

```

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Applicati | State | Target | Target | Application Path |
| ion ID   | on Type   |      | Servers| Clusters|                  |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1,s |          | ${INSTALL_HOME}/myA |
|          |            |        | erver2   |          | pp/myApp.ear         |
+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+
=====
[MASTER]domain1.adminServer>remove-application-target myApp -servers server2
remove server or cluster target from the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicat | Application | State | Target | Target | Application Path |
| ion ID   | Type       |      | Servers| Clusters|                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1 |         | ${INSTALL_HOME}/myA |
|          |            |         |         |         | pp/myApp.ear         |
+-----+-----+-----+-----+-----+-----+
=====
[MASTER]domain1.adminServer>

```



remove-application-target 명령어를 수행할 때 사용할 수 있는 옵션에 대한 자세한 내용은 JEUS Reference 안내서의 "remove-application-target"을 참고한다.

4.3.10. 애플리케이션 정보 확인

본 절에서는 애플리케이션 정보를 확인하는 방법에 대해서 설명한다.

4.3.10.1. 콘솔 툴 사용

콘솔 툴의 **application-info** 명령어를 통해 myApp.ear 애플리케이션에 대한 정보를 조회할 수 있다.

다음은 콘솔 툴에서 애플리케이션 정보를 확인하는 예제이다. 각 옵션에 대한 자세한 설명은 JEUS Reference 안내서의 "application-info"를 참고한다.

```

[MASTER]domain1.adminServer> application-info
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Applicati| Application| State | Server | Cluster | Application Path |
| on ID    | Type      |      | Targets| Targets |                   |
+-----+-----+-----+-----+-----+-----+
| myApp    | EAR        | RUNNING | server1 |         | ${INSTALL_HOME}/myAp |
|          |            |         |         |         | p/myApp.ear         |
+-----+-----+-----+-----+-----+-----+
=====

[MASTER]domain1.adminServer>application-info -id myApp -server server1
Application information for the server [server1] in the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+
| Application| Application Name| Application | State| Server | Cluster |
| ID         |                 | Type       |     | Targets| Targets |
+-----+-----+-----+-----+-----+-----+
| myApp      | myApp          | ear        | RUNNI| server1 |         |
+-----+-----+-----+-----+-----+-----+

```

```

| | | | NG | | |
+-----+-----+-----+-----+-----+
=====

```

```

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail
Application name : myApp
Application [myApp]
=====

```

```

+-----+-----+-----+-----+
| Module Name | Unique Module Name | Module Type |
+-----+-----+-----+-----+
| ejb         | myApp#ejb         | EJB         |
| appclient   | myApp#appclient   | CAR         |
| web         | myApp#web         | WAR         |
+-----+-----+-----+-----+

```

To view detailed information about EJBs or web modules in an EAR, use the "-module" or "-type" option.

```

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail -module ejb
Application name : myApp
General information about the EJB module [ejb].
=====

```

```

+-----+-----+-----+
| Module Name | Unique Module Name |
+-----+-----+-----+
| ejb         | myApp#ejb         |
+-----+-----+-----+
=====

```

Beans

```

+-----+-----+-----+-----+
| Bean Name | Type | Local Export Name | Remote Export Name |
+-----+-----+-----+-----+
| HelloBean | StatelessSessionBean | | |
+-----+-----+-----+-----+
=====

```

```

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail -type war
Application name : myApp
There are no EJBs in this module.
General information about the web module [web].
=====

```

```

+-----+-----+-----+
| Module Name | Unique Module Name | Context Path |
+-----+-----+-----+
| web         | myApp#web         | /hello       |
+-----+-----+-----+
=====

```

Servlets

```

+-----+-----+-----+-----+-----+-----+-----+
| Name | Class | State | Count | Attribute | RegType | URLPatterns |
+-----+-----+-----+-----+-----+-----+-----+
| HelloServlet | dvt.deployment.se | READY | 0 | SYNC | WEB_XML | /HelloServlet |
+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+
=====

Filters
=====
+-----+-----+-----+-----+-----+-----+
| Name | Class | Attribute | RegType | URLPatterns | Servlets |
+-----+-----+-----+-----+-----+-----+
(No data available)
=====

```

```

Listeners
=====
+-----+-----+-----+-----+
| Name | Type | RegType |
+-----+-----+-----+-----+
(No data available)
=====

```

```

EJBs
=====
+-----+-----+-----+-----+-----+-----+
| Bean Name | Type | Local Export Name | Remote Export Name |
+-----+-----+-----+-----+-----+-----+
(No data available)
=====

```

```

[MASTER]domain1.suok>application-info -id myApp -server server1 -detail -module ejb -bean HelloBean
bean HelloBean
Application name : myApp
Module name : ejb
Bean name: HelloBean

```

```

=====
+-----+-----+-----+-----+-----+-----+
| Name | (Count) | WaterMark(High:Low | Bound(Upper:L | Time(Max:Min:T |
| | | :Cur) | ower) | otal) |
+-----+-----+-----+-----+-----+-----+
| create | times(0) | | | |
+-----+-----+-----+-----+-----+-----+
| remove | times(0) | | | |
+-----+-----+-----+-----+-----+-----+
| timed-rb | transaction | | | |
| | n(0) | | | |
+-----+-----+-----+-----+-----+-----+
| request | request(0) | | | |
+-----+-----+-----+-----+-----+-----+
| active-bean | | bean(0:0:0) | | |
+-----+-----+-----+-----+-----+-----+
| rolledback | transaction | | | |
| | n(0) | | | |
+-----+-----+-----+-----+-----+-----+
| total-bean | | bean(0:0:0) | | |
+-----+-----+-----+-----+-----+-----+
| comitted | transaction | | | |
| | n(0) | | | |
+-----+-----+-----+-----+-----+-----+
| MethodReadyCou | | bean(0:0:0) | | |
| nt | | | | |
+-----+-----+-----+-----+-----+-----+

```

```

| active-thread |          | thread(0:0:0) |          |          |
+-----+-----+-----+-----+-----+
| total-remote-t |          | thread(100:100:100) |          |          |
|hread          |          |          |          |          |
+-----+-----+-----+-----+-----+
=====

```

application-info 명령은 도메인에 존재하는 모든 애플리케이션의 정보를 출력한다. 기본적으로 출력되는 정보는 다음과 같다.

항목	설명
Application ID	애플리케이션의 ID이다. 이 값은 도메인에서 유일해야 한다.
Application Type	애플리케이션의 종류로 5가지 중 하나가 될 수 있다. <ul style="list-style-type: none"> ◦ EAR : 애플리케이션 ◦ EJB : EJB 모듈 ◦ WAR : 웹 애플리케이션 모듈 ◦ RAR : 리소스 어댑터 모듈 ◦ CAR : 애플리케이션 클라이언트 모듈
state	애플리케이션이 도메인에서의 상태로 다음 4가지 중 하나가 될 수 있다. <ul style="list-style-type: none"> ◦ INSTALLED ◦ DISTRIBUTED ◦ RUNNING ◦ DEPLOYED <p>각 상태에 대한 자세한 설명은 애플리케이션 상태를 참고한다.</p>
Server Target	애플리케이션이 deploy되어 있는 대상 서버이다.
Cluster Target	애플리케이션이 deploy되어 있는 대상 클러스터이다.
Application Path	Master Server에 존재하는 애플리케이션 파일의 경로를 나타낸다.

4.4. Staging Mode Deploy

exploded 모듈 형태의 애플리케이션은 파일을 압축하여 다른 머신의 MS에 deploy할 수 있다. 이를 Staging Mode Deploy라고 한다. 애플리케이션 파일이 애플리케이션 저장소에 위치하거나 Master Server가 존재하는 머신에 있는 절대 경로를 설정하고 deploy할 수 있다. Staging Mode Deploy 기능에 대한 자세한 내용은 [Staging Mode Deploy](#)를 참고한다.

콘솔 툴 사용

다음은 콘솔 툴을 사용한 Staging Mode Deploy 방법이다. **deploy** 명령어를 staging 옵션을 지정하여 수행한다.


```
[MASTER]domain1.adminServer>deploy exploded -servers server1 -staging
deploy the application for the application [exploded] succeeded.
```

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
```

```
=====
```

Application ID	Application Type	State	Server Targets	Cluster Targets	Application Path
exploded	WAR	RUNNING	server1		/home/user1/apps/exploded
myApp	EAR	RUNNING	server1		\${INSTALL_HOME}/myApp/myApp.ear

```
=====
```

4.5. Deployment Plan을 사용한 Deployment

Deployment Plan은 Deploy Time에 애플리케이션의 DD의 내용을 수정할 수 있는 애플리케이션 외부의 설정 파일이다. JEUS는 XML 형식의 Deployment Plan을 정의하며 EJB, 웹 애플리케이션, EAR의 표준 DD와 JEUS DD에 Deployment Plan을 적용할 수 있도록 지원한다. 애플리케이션을 deploy할 때 Deployment Plan 사용에 대해 명시하면 Deploy Time에 DD와 Deployment Plan의 설정을 merge하여 최종 애플리케이션 설정을 확정된 뒤 Deploy를 수행한다.

본 절에서는 Deployment Plan의 설정 방법과 동작 방식에 대해 설명한다. 또한 deploy 시 Deployment Plan을 적용하는 방법에 대해 설명한다.

4.5.1. Deployment Plan 설정 및 동작 방식

Deployment Plan의 작성 예제를 통해 설정 방법과 동작 방식에 대해 설명한다.

Deployment Plan 설정

다음은 Deployment Plan의 작성 예이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus"
    xmlns:jeus="http://www.tmaxsoft.com/xml/ns/jeus"
    xmlns:jakartaee="https://jakarta.ee/xml/ns/jakartaee">
  <descriptors>
    <!-- For standalone EJB -->
    <descriptor>
      <uri>META-INF/ejb-jar.xml</uri>
      <configurations>
        <configuration>
          <action>REPLACE</action>
          <xpath>//jakartaee:ejb-name[.='ByeBean']</xpath>
          <value>
```

```

        <![CDATA[<ejb-name>HiBean</ejb-name>]]>
    </value>
</configuration>
<configuration>
    <action>DELETE</action>
    <xpath>/child::jakartaee:ejb-jar/child::jakartaee:enterprise-beans/
child::jakartaee:session/child::jakartaee:local-home[.='HelloHomeLocal']</xpath>
</configuration>
<configuration>
    <action>APPEND_CHILD</action>
    <xpath>/child::jakartaee:ejb-jar/descendant::jakartaee:session[2]</xpath>
    <value>
        <![CDATA[<transaction-type>Bean</transaction-type>]]>
    </value>
</configuration>
<configuration>
    <action>INSERT_BEFORE</action>
    <xpath>/jakartaee:ejb-jar/jakartaee:enterprise-beans/jakartaee:session/
jakartaee:ejb-name[.='HelloBean']/../jakartaee:transaction-type</xpath>
    <value>
        <![CDATA[<session-type>Stateless</session-type>]]>
    </value>
</configuration>
</configurations>
</descriptor>
<descriptor>
    <uri>META-INF/jeus-ejb-dd.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:export-name[.='ByeBean']</xpath>
            <value>
                <![CDATA[<export-name>HiBean</export-name>]]>
            </value>
        </configuration>
        <configuration>
            <action>DELETE</action>
            <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:local-export-name[.='HelloBeanLocal']</xpath>
        </configuration>
        <configuration>
            <action>APPEND_CHILD</action>
            <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean</xpath>
            <value>
                <![CDATA[<jeus-rmi>false</jeus-rmi>]]>
            </value>
        </configuration>
        <configuration>
            <action>INSERT_BEFORE</action>
            <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:ejb-name[.='HiBean']/../jeus:jeus-rmi</xpath>
            <value>
                <![CDATA[<thread-max>100</thread-max>]]>
            </value>
        </configuration>
    </configurations>
</descriptor>
<!-- For standalone web application -->

```

```

<descriptor>
  <uri>WEB-INF/web.xml</uri>
  <configurations>
    <configuration>
      <action>REPLACE</action>
      <xpath>/child::jakartaee:web-app/child::jakartaee:servlet-mapping/
child::jakartaee:servlet-name[.='HiServlet']</xpath>
      <value>
        <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
      </value>
    </configuration>
    <configuration>
      <action>DELETE</action>
      <xpath>//jakartaee:load-on-startup</xpath>
    </configuration>
    <configuration>
      <action>APPEND_CHILD</action>
      <xpath>/jakartaee:web-app/descendant::jakartaee:login-config</xpath>
      <value>
        <![CDATA[<auth-method>BASIC</auth-method>]]>
      </value>
    </configuration>
    <configuration>
      <action>INSERT_BEFORE</action>
      <xpath>/jakartaee:web-app/jakartaee:env-entry/
jakartaee:env-entry-value[.='value1']</xpath>
      <value>
        <![CDATA[<env-entry-type>java.lang.String</env-entry-type>]]>
      </value>
    </configuration>
  </configurations>
</descriptor>
<descriptor>
  <uri>WEB-INF/jeus-web-dd.xml</uri>
  <configurations>
    <configuration>
      <action>REPLACE</action>
      <xpath>//jeus:enable-jsp</xpath>
      <value>
        <![CDATA[<enable-jsp>true</enable-jsp>]]>
      </value>
    </configuration>
    <configuration>
      <action>DELETE</action>
      <xpath>/jeus:jeus-web-dd/child::jeus:max-instance-pool-size</xpath>
    </configuration>
    <configuration>
      <action>INSERT_BEFORE</action>
      <xpath>/jeus:jeus-web-dd/descendant::jeus:enable-jsp</xpath>
      <value>
        <![CDATA[<context-path>/hello</context-path>]]>
      </value>
    </configuration>
    <configuration>
      <action>APPEND_CHILD</action>
      <xpath>//jeus:jeus-web-dd</xpath>
      <value>
        <![CDATA[<webinf-first></enabled>false</enabled></webinf-first>]]>
      </value>
    </configuration>
  </configurations>
</descriptor>

```

```

        </configuration>
    </configurations>
</descriptor>
<!-- For EAR -->
<descriptor>
    <uri>META-INF/application.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>/jakartaee:application/jakartaee:library-directory</xpath>
            <value>
                <![CDATA[<library-directory>mylib</library-directory>]]>
            </value>
        </configuration>
    </configurations>
</descriptor>
<descriptor>
    <uri>ejb.jar/META-INF/ejb-jar.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>/jakartaee:ejb-jar/jakartaee:enterprise-beans/
jakartaee:session/jakartaee:ejb-class</xpath>
            <value>
                <![CDATA[<ejb-class>HelloBean</ejb-class>]]>
            </value>
        </configuration>
    </configurations>
</descriptor>
<descriptor>
    <uri>web.war/WEB-INF/web.xml</uri>
    <configurations>
        <configuration>
            <action>REPLACE</action>
            <xpath>/jakartaee:web-app/jakartaee:servlet-mapping/
jakartaee:servlet-name</xpath>
            <value>
                <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
            </value>
        </configuration>
    </configurations>
</descriptor>
</descriptors>
</jeus-deployment-plan>

```

다음은 <descriptor>의 하위 태그에 대한 설명이다.

- <descriptor>

<descriptor>는 여러 <configuration> 태그로 이루어져 있다.

- <configuration>

<configuration>은 target DD에 적용할 하나의 설정 단위로 하위에 <action>, <xpath>, <value> 태그로 구성된다.

태그	설명
<action>	<p>DD의 특정 태그에 대해서 어떤 설정을 변경할 것인지를 명시한다.</p> <p>다음은 설정값에 대한 설명이다.</p> <ul style="list-style-type: none"> ◦ DELETE : DD의 특정 태그를 삭제한다. ◦ REPLACE : DD의 특정 태그를 임의의 태그로 대체한다. ◦ APPEND_CHILD : DD의 특정 태그의 last child로써 임의의 태그를 추가한다. ◦ INSERT_BEFORE : DD의 특정 태그의 previous sibling으로서 임의의 태그를 삽입한다.
<xpath>	<p>DD의 특정 태그를 xpath expression으로 지정한다. 이때 <xpath> 표준에 따라 <xpath> 경로에 위치하는 모든 태그는 XML Name Space를 통해 qualified name으로 표현되어야 한다. 이를 위해 Deployment Plan에 각 DD의 XML Name Space를 Name Space Prefix와 함께 선언한다.</p> <p>예를 들어 JEUS DD에 있는 임의의 태그를 <xpath>로 지정할 경우 <xpath> 경로에 위치하는 모든 태그는 JEUS XML Name Space(http://www.tmaxsoft.com/xml/ns/jeus)와 함께 qualified name으로 표현될 필요가 있다. Deployment Plan에서는 JEUS XML Name Space를 'jeus' Prefix로 매칭시켜 놓았으므로 <xpath> 경로에 위치하는 모든 태그의 이름 앞에 'jeus' Prefix를 붙여 qualified name을 표현한다.</p>
<value>	<p><value>는 <action> 값이 REPLACE, APPEND_CHILD, INSERT_BEFORE인 경우에만 유효하다.</p> <p><action> 값에 따른 설정값은 다음과 같다.</p> <ul style="list-style-type: none"> ◦ REPLACE : <xpath>로 지정한 태그를 대체할 새로운 태그를 설정한다. ◦ APPEND_CHILD : <xpath>로 지정한 태그에 last child로 추가할 태그를 설정한다. ◦ INSERT_BEFORE : <xpath>로 지정한 태그의 previous sibling으로 삽입할 태그를 설정한다. <p><value> 값으로 설정되는 태그는 일반적으로 depth를 갖는 XML fragment 형태를 가정하므로 <value> 값은 특별히 CDATA section으로 표현한다. CDATA section에 action 수행을 위해 필요한 XML fragment를 그대로 적는다.</p>

Deployment Plan 동작 방식

위의 작성 예제를 바탕으로 태그별 동작 방식에 대해 설명한다.

• <descriptor>

Deployment Plan은 여러 개의 <descriptor> 태그로 이루어져 있다.

<descriptor> 단위로 target이 되는 DD를 지정하며 이를 위해 <uri> 태그를 사용한다. 애플리케이션 파일을 루트(root)로 가정하고 그로부터 정해지는 DD의 상대 경로를 <uri> 값으로 명시하여 target DD를 결정한다.

- 예를 들어 Standalone EJB 모듈의 표준 DD는 애플리케이션 루트로부터 항상 META-INF/ejb-jar.xml에

위치하며 JEUS DD는 META-INF/jeus-ejb-dd.xml에 위치한다. 그러므로 위의 Deployment Plan의 첫 번째 <descriptor>는 그것의 <uri> 값으로부터 Standalone EJB 모듈의 표준 DD를 위한 것이고 두 번째 <descriptor>는 그것의 <uri> 값으로부터 Standalone EJB 모듈의 JEUS DD를 위한 것임을 알 수 있다. Standalone 웹 애플리케이션이나 EAR에 대해서도 동일한 규칙이 적용된다.

- <uri> 값이 'ejb.jar/META-INF/ejb-jar.xml'인 <descriptor>는 EAR에 속한 EJB 모듈(파일 이름이 ejb.jar)을 위한 것이고, <uri> 값이 'web.war/WEB-INF/web.xml'인 <descriptor>는 EAR에 속한 WEB 모듈(파일 이름이 web.war)을 위한 것이다.
- 하나의 Deployment Plan은 여러 개의 서로 다른 애플리케이션 Deploy를 위해 사용될 수 있다. deploy하는 애플리케이션의 DD와 <uri> 값이 매칭되어 유효한 것으로 판단되는 <descriptor>만 선택되어 해당 애플리케이션을 deploy할 때 적용되는 방식이므로 그 외의 <descriptor>는 Deploy에 전혀 영향을 주지 않는다.

- <configuration>

다음은 첫 번째 <descriptor> 설정을 예로 들어 설명한 것으로 <descriptor>의 설정에 의해 DD에 실제로 수정되는 사항에 대한 설명이다.

- 첫 번째 <descriptor>는 앞서 언급했듯이 Standalone EJB 모듈의 표준 DD를 조작한다. 가정 먼저 수행되는 조작 action은 특정 태그의 대체로써 첫 번째 <configuration>으로 표현되어 있다. 값이 'ByeBean'이라는 <ejb-name> 태그를 값이 'HiBean'인 <ejb-name> 태그로 대체하고 있음을 확인할 수 있다.
- 두 번째 <configuration>은 값이 'HelloHomeLocal'인 <local-home> 태그의 삭제를 표현한다.
- 세 번째 <configuration>은 두 번째 <session> 태그의 last child로써 <transaction> 태그를 추가함을 표현한다.
- 네 번째 <configuration>은 <ejb-name> 태그의 값이 'HelloBean'인 <session> 태그의 previous sibling으로써 <session-type> 태그를 삽입함을 표현한다.

4.5.2. Deployment Plan Install

Deployment Plan을 이용하여 deploy를 하기 위해서는 우선 Deployment Plan을 도메인에 install해야 한다. 애플리케이션과 유사하게 Deploy에 이용할 수 있는 Deployment Plan 또한 도메인에 install된 것들에 한한다. Deployment Plan을 install할 때는 도메인에서의 Deployment Plan의 식별자를 설정할 수 있다.

Deployment Plan은 콘솔 툴을 사용해서 install할 수 있다.

콘솔 툴 사용

콘솔 툴에서 **install-deployment-plan** 명령어를 통해 Deployment Plan을 install할 수 있다.

```
[MASTER]domain1.adminServer>install-deployment-plan -path /home/user1/plans/jeus-deployment-plan.xml
-name plan1
Installing the deployment plan [plan1] was successful.
```



install-deployment-plan 명령의 사용 방법에 대한 자세한 내용은 JEUS Reference

안내서의 "install-deployment-plan"을 참고한다.

4.5.3. Install한 Deployment Plan 확인

콘솔 툴을 사용해서 Install한 Deployment Plan을 확인할 수 있다. 각각의 Deployment Plan은 install할 때 설정한 Deployment Plan의 식별자로 구분되며 Deployment Plan별로 그것이 적용된 애플리케이션 목록이 표시된다. Deployment Plan의 실제 파일 내용도 화면에 출력할 수 있다.

콘솔 툴 사용

콘솔 툴에서 **deployment-plan-info** 명령어를 통해서 도메인에 install한 Deployment Plan이 적용된 애플리케이션의 목록을 조회할 수 있고, 특히 특정 Deployment Plan을 지정하면 해당 Deployment Plan의 내용을 화면에서 확인할 수 있다.

- Deployment Plan의 목록 조회

```
[MASTER]domain1.adminServer>deployment-plan-info
The list of deployment plans installed in the domain and the applications to which each
deployment plan applies
=====
+-----+-----+
| Deployment plan | Applications |
+-----+-----+
| plan1          |              |
+-----+-----+
=====
```

- 특정 Deployment Plan의 내용 조회

```
[MASTER]domain1.adminServer>deployment-plan-info -name plan1
<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
  ...
</jeus-deployment-plan>
```



deployment-plan-info 명령 사용 방법에 대한 자세한 내용은 JEUS Reference 안내서의 "deployment-plan-info"를 참고한다.

4.5.4. Deployment Plan을 적용한 Deploy

콘솔 툴을 사용해서 Install한 Deployment Plan을 적용하여 Deploy를 수행할 수 있다.

콘솔 툴 사용

콘솔 툴에서 **deploy-application** 명령어를 통해 Deployment Plan을 사용한 Deploy를 수행할 수 있다.

```
[MASTER]domain1.adminServer>deploy-application webapp -all -plan plan1
deploy the application for the application [webapp] succeeded.
```

4.5.5. 애플리케이션에 적용된 Deployment Plan 확인

콘솔 툴을 사용해서 임의의 애플리케이션에 어떤 Deployment Plan이 적용되었는지 확인할 수 있다.

콘솔 툴 사용

콘솔 툴에서 **application-info** 명령어를 통해 애플리케이션에 적용된 Deployment Plan을 확인할 수 있다.

```
[MASTER]domain1.adminServer>application-info -id webapp -detail
Application information for the domain [domain1].
=====
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Appli| Applic|State|Server|Cluster|    Running    | Applicati| Applicati| Plan|
| cation| ation |      |Target|Targets| Servers      | on Path  | on Time  | Name |
| ID   | Type  |      |s      |        |              |          |          |      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|webapp| WAR   | RUNN| ALL   | ALL   | server1,serve| ${INSTALL| Tue May | plan1|
|      |      | ING |       |       | r2,server3,adm| HOME}/web| 28    |      |
|      |      |     |       |       | inServer    | app/deploy|22:45:13 |      |
|      |      |     |       |       | ment_plan  | KST 2013  |      |
|      |      |     |       |       |            | web.war   |      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
=====
```

4.5.6. Deployment Plan Uninstall

콘솔 툴을 사용해서 Deployment Plan을 도메인에서 uninstall할 수 있다. Uninstall된 Deployment Plan은 도메인에서 더 이상 유효하지 않으며 당연히 deploy할 때도 사용할 수 없다.

콘솔 툴 사용

콘솔 툴에서 **uninstall-deployment-plan** 명령어를 통해 Deployment Plan을 uninstall할 수 있다.

```
[MASTER]domain1.adminServer>uninstall-deployment-plan plan1
Uninstalling the deployment plan was successful.
```



uninstall-deployment-plan 명령의 사용 방법에 대한 자세한 내용은 JEUS Reference

안내서의 "uninstall-deployment-plan"을 참고한다.

4.5.7. Deployment Plan Redeploy

Deployment Plan을 이용해서 deploy한 애플리케이션은 redeploy할 때 기본적으로 기존의 Deployment Plan이 그대로 적용된다. 그러나 redeploy할 때 Deployment Plan을 새롭게 설정하면 새로운 Deployment Plan을 적용하여 redeploy할 수 있다.