

## 하나씩 쉽게 따라 해보는 IBM WebSphere Application Server(WAS) v7 – 24

이정운 ([juwlee@kr.ibm.com](mailto:juwlee@kr.ibm.com))

하나씩 쉽게 따라 해보는 IBM WAS v7 스물 네 번째 이야기를 시작합니다. 스물 네 번째 이야기는 3부 강의의 세번째 로서 wsadmin 스크립트에 대해서 배워보는 시간을 가져보도록 하겠습니다.

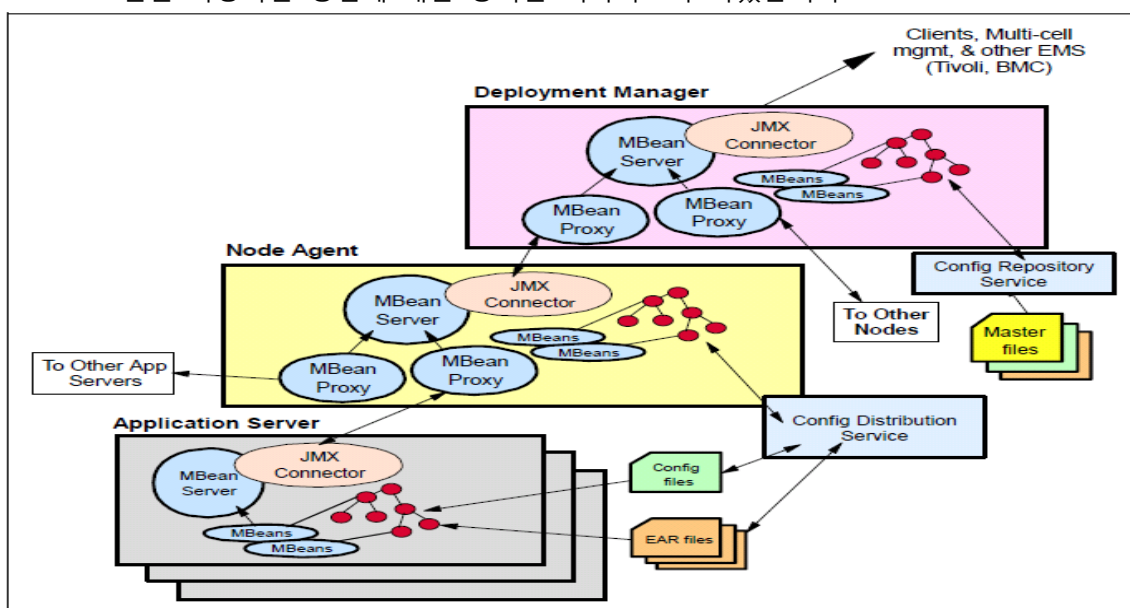
IBM WAS 는 웹 브라우저를 이용한 관리콘솔 말고도 wsadmin 이라는 script 툴을 제공하여 전체 WAS 제어 및 관리를 할 수 있도록 돕고 있습니다. 일반 관리자의 경우는 웹 브라우저를 이용한 관리콘솔을 사용하는 것이 편리하지만, 좀 더 관리에 익숙해지고 나면 wsadmin 툴을 이용해서 script 를 작성해두고 재사용하는 것이 보다 빠르고 편리한 관리를 가능하게 해 줄 수 있습니다. (관리자 선호도와 편의에 의해서 선택적으로 사용가능합니다.)

Wsadmin 툴의 내부를 조금 더 살펴보면 JMX(Java Management eXtensions) API 를 이용하여 MBean 객체를 통해 WAS 의 각 설정들을 관리하는 도구입니다. 여기서 JMX 는 자바 기반의 어플리케이션을 모니터링하고 관리하는 기능을 제공하는 표준으로서 사전을 통해서 간단히 살펴보면 아래와 같습니다.

JMX(Java Management eXtensions)는 응용 프로그램 소프트웨어/객체/장치 (프린터 등) 및 서비스 지향 네트워크 등을 감시 관리를 위한 도구를 제공하기 위한 자바 API이다. 이러한 리소스는 MBean(Managed Bean)라는 객체로 표현된다.

참조: 한국어 위키피디아

따라서 wsadmin 을 잘 이해하고 나면 외부에서 접속가능한 본인만의 툴을 커스터마이징하여 만들수도 있다는 점, 잘 유의하시기 바라겠습니다. 그럼 지금부터 좀 더 발전된 관리를 위한 wsadmin 툴을 사용하는 방법에 대한 강좌를 시작하도록 하겠습니다.

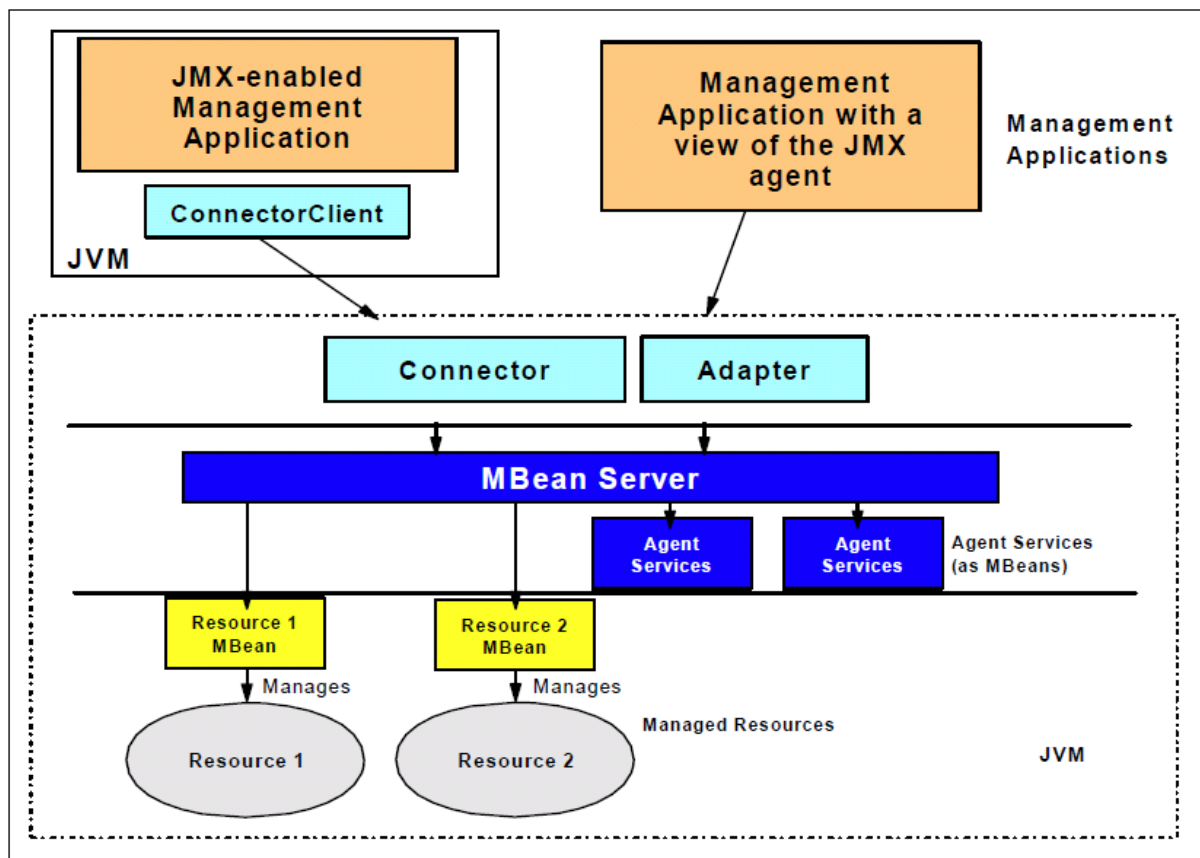


참조: IBM WAS v7 Redbook

## Part 1. Wsadmin 툴 기본 이해하기

Wsadmin 툴에 대해서 먼저 간단히 소개하자면 script 를 처리할 수 있는 툴입니다. 그렇다고 모든 script 를 처리할 수 있는 것은 아니고 Jacl 과 Jython, 두 가지 script 를 처리하는 툴입니다. 중요한 것은 이러한 script 를 이용해서 WAS 에서 구동 중인 MBean 객체와 통신할 수 있다 라는 것 입니다. Script 를 이용해서 JMX 표준 객체인 MBean 과 통신하여 WAS 의 관리 객체들을 이용해서 단순하고 표준화된 자바 객체의 관리를 할 수 있습니다. 다시 말하여, 관리콘솔을 통해서 WAS 를 제어, 관리하는 것이 아니라 script 를 이용하여 WAS 를 제어, 관리해줄 수 있는 툴 입니다.

이미 강좌 처음에 언급드렸지만 WAS 는 JMX 의 MBean 객체를 이용해서 관리 가능하며 하단의 그림과 같이 되어 있는 환경에서 wsadmin 은 툴이라고 말을 하였지만 실질적으로는 Connector 를 통해서 MBean 서버에 접속할 수 있는 JMX agent 역할을 하고 있습니다. 즉, Jython 으로 작성된 script 명령들을 MBean 서버로 전송하여 수행하고 그 결과를 보여줍니다.



참조: IBM WAS v7 Redbook

다음으로 Jacl 은 Java 코드를 이용해서 TCL 을 대체 구현한 script 언어입니다. (TCL 은 스크립트 언어로써 비교적 배우기 쉽고 충분히 강력한 기능을 제공합니다. 보통 빠른 프로토타이핑, 스크립트 프로그램, GUI 및 테스팅에 많이 사용됩니다.) 그러나 Jacl 은 하단에 IBM이 발표한 것 처럼 WAS 에서 지원은 되지만 향후 Jython 에 집중하여 투자할 것이기 때문에 본 강좌에서는 생략하도록 하겠습니다.

The Jacl language has been stabilized in Version 7 of the product. IBM® does not currently plan to deprecate or remove this capability in a subsequent release of the product; but future investment will be focused on the Jython language, which is the strategic alternative. You do not need to change any of your existing applications and scripts that use Jacl; but you should consider using the strategic alternative for new applications.

참조: IBM WAS v7 Information Center

Jython 은 많은 사람들이 잘 아시는 script 언어인 Python 의 Java 버전입니다. Jython 의 경우도 script 이긴 하지만 프로그래밍 언어입니다. If, for, while 같은 문장을 사용할 수 있으며 함수를 선언하고 사용하는 것도 가능합니다. 심지어는 Java class 를 import 할 수도 있는 강력한 script 언어입니다.

Wsadmin 은 Jython 으로 작성된 script 를 통하여 다음과 같은 WAS 의 자바 객체들을 제어할 수 있으며 해당 객체들을 이용해서 WAS 를 제어, 관리 할 수 있습니다.

# **AdminControl**: 작동 명령을 실행할 경우에 사용합니다.

# **AdminConfig**: WebSphere Application Server 컴포넌트를 작성하거나 수정하기 위해 구성 명령을 실행할 경우에 사용합니다.

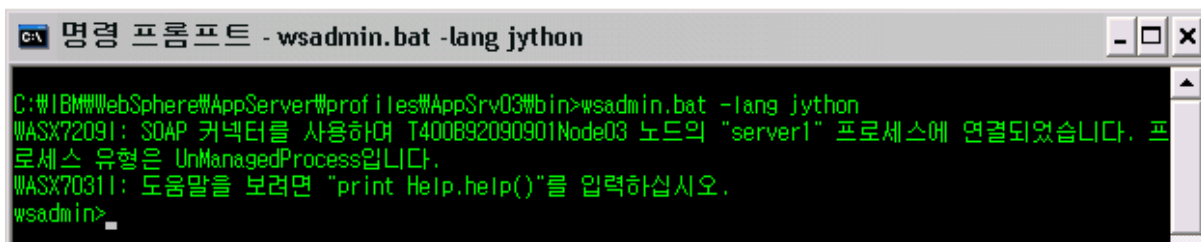
# **AdminApp**: 응용프로그램을 관리할 경우에 사용합니다.

# **AdminTask**: 관리 명령을 실행할 경우에 사용합니다.

# **Help**: 일반 도움말을 볼 경우에 사용합니다.

백문이 불여일견이라는 말이 있듯이 한번 그냥 실행해 보도록 하겠습니다. Wsadmin 툴을 실행하는 것은 아주 간단해서 WAS\_ROOT/bin 폴더에 있는wsadmin.bat/sh 파일을 실행시키기만 하면 됩니다. 다만 이번 강좌에서는 Jython 을 사용할 것이기 때문에 jython 언어를 사용하겠다고 옵션을 지정하여 wsadmin 툴을 시작합니다. (예: wsadmin.bat -lang jython) (관리보안을 설정하신 경우에는 하단과 같이 ID 와 Password 를 물어보는 창이 나오는데 WAS 관리콘솔에 들어갈 때 사용하는 ID 와 Password 를 입력하면 됩니다.)

그러면 하단과 같이 wsadmin 툴이 실행된 화면을 확인할 수 있습니다.

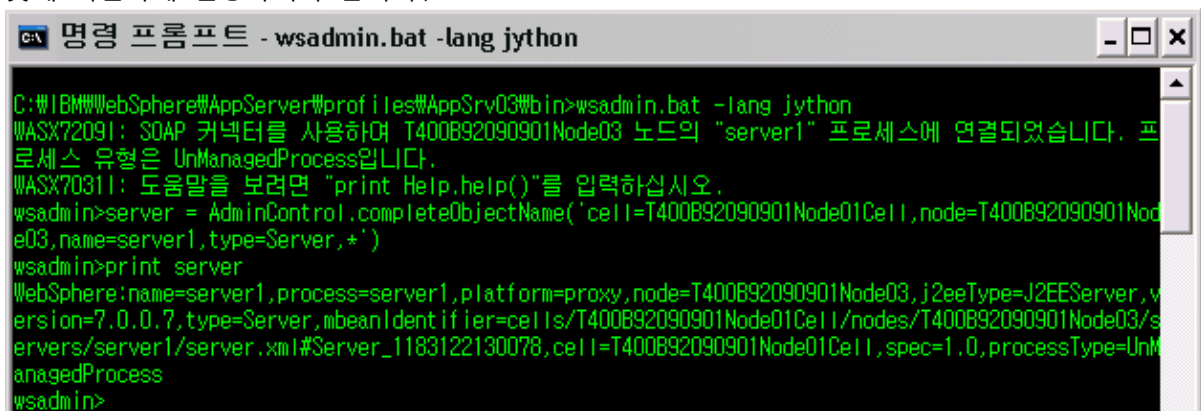


```
C:\IBM\WebSphere\AppServer\profiles\AppSrv03\bin>wsadmin.bat -lang jython
WSAX72091: SOAP 커넥터를 사용하여 T400B92090901Node03 노드의 "server1" 프로세스에 연결되었습니다. 프로세스 유형은 UnManagedProcess입니다.
WSAX70311: 도움말을 보려면 "print Help.help()"를 입력하십시오.
wsadmin>
```

실제로 wsadmin 의 Jython script 를 활용해 보기 위하여 다음과 같은 명령을 명령 프롬프트에 순차적으로 입력합니다.

```
server
AdminControl.completeObjectName('cell=T400B92090901Node01Cell,node=T400B92090901Node03,name=server1,type=Server,#')
print server
```

그러면 하단과 같이 현재 운영중인 WAS 서버인 server1 에 대한 상태 정보를 하던처럼 확인할 수 있습니다. (단, 위의 Jython script 에서 cell 이름과 node 이름, server 이름은 본인의 WAS 에 맞게 적절하게 변경하셔야 합니다.)



```
C:\IBM\WebSphere\AppServer\profiles\AppSrv03\bin>wsadmin.bat -lang jython
WSAX72091: SOAP 커넥터를 사용하여 T400B92090901Node03 노드의 "server1" 프로세스에 연결되었습니다. 프로세스 유형은 UnManagedProcess입니다.
WSAX70311: 도움말을 보려면 "print Help.help()"를 입력하십시오.
wsadmin>server = AdminControl.completeObjectName('cell=T400B92090901Node01Cell,node=T400B92090901Node03,name=server1,type=Server,#')
wsadmin>print server
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,version=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=UnManagedProcess
wsadmin>
```

그런데 한가지 이상한 점을 느끼셨나요? 그것은 wsadmin 이 script 툴이라고 했는데, script 툴처럼 보이지 않고 한줄씩 명령을 실행하는 명령어 입력기 같은 기분이 드셨을 것입니다. 이것은 wsadmin 툴의 명령어 입력모드로 들어와서 실행했기 때문에 이처럼 보이는 것이고 실제 script 파일을 만들어서 위와 동일한 작업을 해보도록 하겠습니다.

먼저 ServerStatus.py 라는 텍스트 파일을 만들고 위에서 입력한 내용을 그대로 입력합니다. 그 후 wsadmin.bat -lang jython -f ServerStatus.py -user wasadm -password wasadm 을 입력합니다. 그러면 다음과 같이 이전에서 봤던 동일한 결과를 하단처럼 확인할 수 있습니다. (여기서 -user 옵션과 -password 옵션은 관리보안 때문에 새창이 떠서 ID 와 password 를 물어보는 것을 미리 입력하는 옵션입니다.)

```
C:\IBM\WebSphere\AppServer\profiles\AppSrv03\bin>wsadmin.bat -lang jython -f ServerStatus.py -user wasadm -password wasadm
WASX7209I: SOAP 커넥터를 사용하여 T400B92090901Node03 노드의 "server1" 프로세스에 연결되었습니다. 프로세스 유형은 UnManagedProcess입니다.
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,version=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=UnManagedProcess

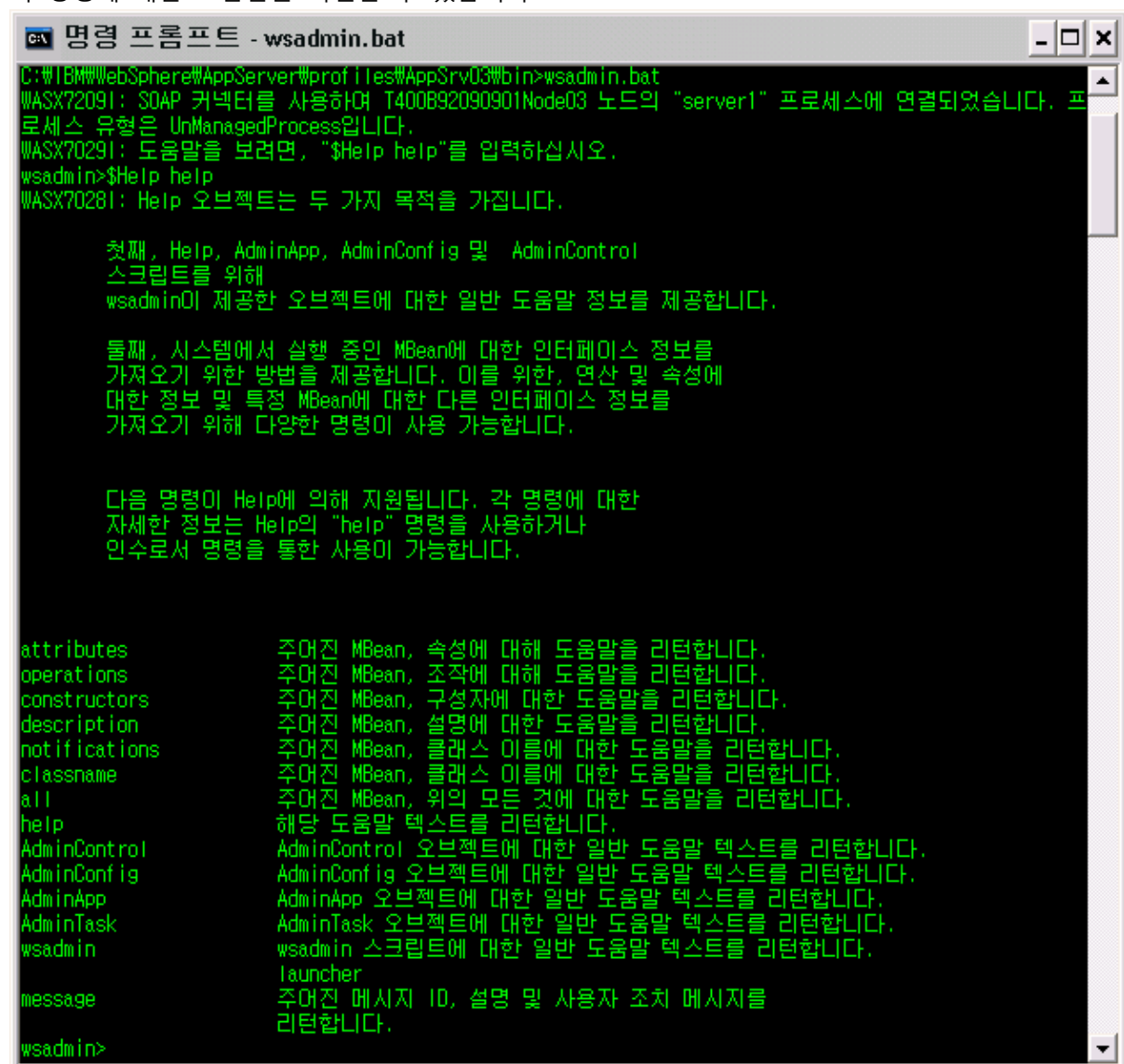
C:\IBM\WebSphere\AppServer\profiles\AppSrv03\bin>
```

어때요? Wsadmin 에서 이제 좀더 script 툴 같은 면모가 느껴지시나요?

## Part 2. Wsadmin 툴 심화

지금까지는 Wsadmin 툴의 기본을 살펴봤고 간단하지만 실제로 구동되는 모습도 확인해 봤습니다. Wsadmin 에 사용되는 Jython 자체가 프로그래밍 언어이며 wsadmin 을 통해서 WAS 에 관련된 모든 작업이 가능하기 때문에 이 강좌에서 전체적인 설명을 다루는 것은 무리이기 때문에 그중 위에서 언급했던 중요한 몇가지 객체들과 샘플들을 살펴보고 조금 더 이해할 수 있는 시간을 가져보도록 하겠습니다. (좀 더 자세한 각 명령어 별 작동 방법과 예제는 IBM WAS v7 Information Center 에 자세히 나오니 해당 자료를 참고하시기 바라겠습니다.)

우선 가장 이해가 빠르고 단순한 Help 객체부터 보도록 하겠습니다. 하단처럼 \$Help help 를 입력하면 Help 명령사용에 대한 도움말을 확인해 보실 수 있습니다. 본 명령을 이용해서 다른 객체와 명령에 대한 도움말을 확인할 수 있습니다.



```
C:\IBM\WebSphere\AppServer\profiles\AppSrv03\bin>wsadmin.bat
WASX7209I: SOAP 커넥터를 사용하여 T400B92090901Node03 노드의 "server1" 프로세스에 연결되었습니다. 프로세스 유형은 UnManagedProcess입니다.
WASX7209I: 도움말을 보려면, "$Help help"를 입력하십시오.
wsadmin>$Help help
WASX7208I: Help 오브젝트는 두 가지 목적을 가집니다.

첫째, Help, AdminApp, AdminConfig 및 AdminControl
스크립트를 위해
wsadmin이 제공하는 오브젝트에 대한 일반 도움말 정보를 제공합니다.

둘째, 시스템에서 실행 중인 MBean에 대한 인터페이스 정보를
가져오기 위한 방법을 제공합니다. 이를 위한, 연산 및 속성에
대한 정보 및 특정 MBean에 대한 다른 인터페이스 정보를
가져오기 위해 다양한 명령이 사용 가능합니다.

다음 명령이 Help에 의해 지원됩니다. 각 명령에 대한
자세한 정보는 Help의 "help" 명령을 사용하거나
인수로서 명령을 통한 사용이 가능합니다.

attributes      주어진 MBean, 속성에 대해 도움말을 리턴합니다.
operations      주어진 MBean, 조작에 대해 도움말을 리턴합니다.
constructors     주어진 MBean, 구성자에 대한 도움말을 리턴합니다.
description      주어진 MBean, 설명에 대한 도움말을 리턴합니다.
notifications    주어진 MBean, 클래스 이름에 대한 도움말을 리턴합니다.
classname        주어진 MBean, 클래스 이름에 대한 도움말을 리턴합니다.
all              주어진 MBean, 위의 모든 것에 대한 도움말을 리턴합니다.
help            해당 도움말 텍스트를 리턴합니다.
AdminControl     AdminControl 오브젝트에 대한 일반 도움말 텍스트를 리턴합니다.
AdminConfig      AdminConfig 오브젝트에 대한 일반 도움말 텍스트를 리턴합니다.
AdminApp         AdminApp 오브젝트에 대한 일반 도움말 텍스트를 리턴합니다.
AdminTask        AdminTask 오브젝트에 대한 일반 도움말 텍스트를 리턴합니다.
wsadmin         wsadmin 스크립트에 대한 일반 도움말 텍스트를 리턴합니다.
launcher        주어진 MBean, 속성에 대해 도움말을 리턴합니다.
message          주어진 메시지 ID, 설명 및 사용자 조치 메시지를
리턴합니다.
wsadmin>
```

다음으로는 AdminControl 객체에 대한 좀 더 자세한 설명을 드리도록 하겠습니다. AdminControl 은 WAS 서버의 객체를 관리하기 위한 작동 명령을 실행할 경우에 사용됩니다. JMX에 의해 지정된 매개변수를 사용하거나 매개변수에 대한 문자열을 사용하여 호출할 수 있습니다. 이러한 조작 명령 외에도, AdminControl 오브젝트는 추적, 서버와 다시 연결 및 데이터 유형 변환을 위한 유틸리티 명령을 지원합니다.

# completeObjectName	# invoke
# getAttribute	# invoke_jmx
# getAttribute_jmx	# isRegistered
# getAttributes	# isRegistered_jmx
# getAttributes_jmx	# makeObjectName
# getCell	# queryMBeans
# getConfigId	# queryNames
# getDefaultDomain	# queryNames_jmx
# getDomainName	# reconnect
# getHost	# setAttribute
# getMBeanCount	# setAttribute_jmx
# getMBeanInfo_jmx	# setAttributes
# getNode	# setAttributes_jmx
# getObjectInstance	# startServer
# getPort	# stopServer
# getPropertiesForDataSource (Deprecated)	# testConnection
# getType	# trace
# help	

AdminControl.help() 명령을 입력하면 각 명령에 대한 간단한 도움말이 나오므로 처음 사용하시는 분들을 해당 명령을 이용해서 도움말을 확인해보도록 합니다.

```
wsadmin>print AdminControl.help()
WASX7027I: AdminControl 오브젝트는 WebSphere
서버 프로세스에서 실행되는 MBean의 조작을 가능하게 합니다. 스크립트
클라이언트에서 사용 가능한 MBean의 수와 유형은 클라이언트가 연결되는
서버에 따라 다릅니다. 클라이언트가 Deployment Manager에 연결될 경우,
Deployment Manager에서 실행 중인 모든 MBean, Deployment Manager에
연결된 Node Agent에서 실행 중인 모든 MBean, 그리고 이러한 노드에
있는 응용프로그램 서버에서 실행 중인 모든 MBean이

다음은 AdminControl에서 지원되는 명령입니다.
이러한 명령에 대한 자세한 정보를 보려면, AdminControl의
"help" 명령을 사용하고 인수로서 각 명령의 이름을
입력하십시오.

이러한 명령 중 다수는 서로 다른 두 개의 서명 세트를 지원합니다.
하나는 문자열을 승인하고 리턴하는 서명 세트이고, 다른 하나는
JMX 오브젝트(예: ObjectName 및 AttributeList)에서 사용되는 하위 레벨 세트입니다.
대부분의 경우, 문자열 서명이 보다 유용하게 사용됩니다.
그러나 JMX-오브젝트 서명 버전 또한 제공됩니다. 각각의
JMX-오브젝트 서명 명령은 명령 이름에 "_jmx"가 추가됩니다.
그러므로 "invoke_jmx" 명령뿐 아니라 "invoke" 명령도 있습니다.

completeObjectName
    템플릿 이름이 주어진 오브젝트 이름의 문자열 버전을
    리턴합니다.
```

AdminControl 객체도 굉장히 많은 명령을 가지고 있어서 이번 강좌에서 다 확인하기는 어려우며 대표적으로 자주 사용될 만한 명령을 소개하자면 completeObjectName, getAttribute, setAttribute, startServer, stopServer, invoke 등입니다

팁1 : print AdminControl.help('명령어') 를 입력하면 하단처럼 해당 명령의 인수라던가 사용방법, 설명등을 하단의 그림과 같이 확인할 수 있습니다. 이는 AdminApp 나 AdminConfig 같은 다른 객체도 동일하므로 사용방법을 모르시면 help 를 이용한 도움말을 통해서 사용방법을 가이드 받을 수 있습니다.

```
wsadmin>print AdminControl.help('invoke')
WASX7047I: 메소드: invoke

인수: object name, operation

설명: "object name"에 의해 설명된 MBean의 "operation"에 의해 이를 지정된 연산을 호출합니다. 연산에 대해 전달된 인수가 없습니다.

메소드: invoke

인수: object name, operation, arguments

설명: "arguments"에 지정된 매개변수를 사용하며 "object name"에 지정된 MBean에 대한 "operation"에 이를 지정된 조작을 호출합니다. 이 조작에 인수가 필요하지 않으면, "arguments" 매개변수를 생략해도 됩니다.

메소드: invoke

인수: object name, operation, arguments, signature

설명: "arguments"에 지정된 매개변수와 "signature"에 지정된 서명을 사용하며 "object name"에 지정된 MBean에 대한 "operation" 조작에 이를 지정된 조작을 호출합니다.
wsadmin>
```

completeObjectName 은 아마도 wsadmin 을 사용할 경우에 가장 많이 사용하는 명령어로서 완벽한 객체이름을 만들 수도록 도와주는 명령어 입니다. Jython 을 이용해서 작업을 할 경우에 가장 중요하면서도 먼저 진행해야 하는 것은 본인이 작업할 객체를 아는 것입니다. 작업할 객체를 알아야 그 객체에 대한 실질적인 작업이 가능합니다. 파트 1 에서 보여드렸던 예제가 여기서도 좋은 예제가 될 수 있습니다.

```
server =
AdminControl.completeObjectName('cell=T400B92090901Node01Cell,node=T400B92090901Node03,name=server1,type=Server,#')
```

지금 운영중인 server1 이라는 WAS 서버에 대한 객체의 이름(fullname) 을 받아오는 것입니다. 다른 예를 하나 더 살펴보도록 하겠습니다.

```
objNameString = AdminControl.completeObjectName("WebSphere:type=Server,#")
```



위와 같이 입력하면 Server 타입의 객체의 이름을 objNameString 으로 받아오는 것이며 print 문을 이용해서 출력해보면 맨 위의 예제와 동일한 결과를 얻을 수 있습니다. (두 예제의 차이는 광범위하게 Server 로 받아오느냐, cell 과 node 까지 입력하여 지정된 server 의 객체 이름을 받아오느냐의 차이입니다. 지금 예제가 돌아가는 환경에서는 server1 이라는 WAS 서버가 하나 밖에 없어서 동일한 결과가 나오지만 여러 WAS 서버가 있는 환경이라면 결과가 달라질 수 있습니다.)

```
wsadmin>objNameString = AdminControl.completeObjectName('WebSphere:type=Server,*)
wsadmin>print objNameString
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,version=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=UnManagedProcess
wsadmin>
```

다음으로 getAttribute 와 setAttribute 는 이름 그대로 해당 객체의 특정 속성(attribute) 의 값을 받아오거나 설정할 때 사용됩니다. 예제를 돌려보시면 바로 이해하실 수 있을텐데, 위의 예제와 연결하여 print AdminControl.getAttribute(objNameString, 'name') 을 입력하면 하단처럼 'server1' 이라는 속성 값을 출력할 수 있습니다.

```
wsadmin>objNameString = AdminControl.completeObjectName('WebSphere:type=Server,*)
wsadmin>print objNameString
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,version=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=UnManagedProcess
wsadmin>print AdminControl.getAttribute(objNameString, 'name')
server1
wsadmin>
```

startServer 와 stopServer 도 이름과 동일한 기능을 제공합니다. AdminControl.stopServer('server1', 'Node 명') 와 같은 방식으로 해당 WAS 서버를 중지, 시작 가능합니다. 단, 하단의 실행 결과를 보시면 아시겠지만 'UnManagedProcess' 인 경우에는 시작이 불가능합니다. 'UnManagedProcess' 는 WAS ND 환경의 Dmgr 의 관리를 받지 않는 WAS 서버나 웹서버를 의미합니다. 간단한 예제를 보여드리기 위하여 현재 샘플을 테스트 하는 환경이 server1 이라는 WAS 서버만 단독으로 운영하여 사용하는 환경이라 아래와 같은 결과를 받을 수 있는 것입니다. WAS ND 를 활용하여 Dmgr 과 Node 를 운영하는 환경에서는 사용이 가능한 명령입니다.

```
wsadmin>AdminControl.stopServer('server1', 'T400B92090901Node03')
WASX7337I: "server1" 서버에 대한 중지가 호출됨. 중지 완료 대기 중.
U'WASX7264I: #uC911#uC9C0#uAC00 "T400B92090901Node03" #uB178#uB4DC#uC5D0#uC11C "server1" #uC11C#uBCB4#uC5D0 #uB300#uD574 #uC644#uB8CC#uB418#uC5C8#uC2B5#uB2C8#uB2E4.'
wsadmin>
wsadmin>
wsadmin>AdminControl.startServer('server1', 'T400B92090901Node03')
WASX7015E: 명령 실행 중 예외: "AdminControl.startServer('server1', 'T400B92090901Node03')". 예외 정보:
com.ibm.ws.scripting.ScriptingException: WASX7254E: "startServer" 조치는 프로세스 유형이 "UnManagedProcess"인 경우, 지원되지 않습니다.
wsadmin>
```

AdminControl 객체에 대해 마지막으로 설명드릴 명령어는 invoke 입니다. invoke 명령을 사용하여 매개변수가 포함되거나 포함되지 않은 오브젝트 조작을 호출할 수 있으며 제공한 매개변수 목록을 사용하여 오브젝트 조작을 호출할 수 있습니다. (MBean 메소드가 동기인 경우, 조작이 완료

되었을 때에만 제어가 wsadmin 도구로 리턴되며 Mbean 메소드가 비동기인 경우, 제어는 호출된  
타스크가 완료되지 않았더라도 즉시 wsadmin 도구로 리턴됩니다.)

Invoke 명령어는 상당히 중요한 명령입니다. 이 명령어가 중요한 이유는 이름 그대로 무언가를  
작동시킬 수 있기 때문입니다. 무언가라고 하는 것은 MBean 객체가 가지고 있는 필드나 메소드  
입니다.

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.javadoc.doc/web/mbeanDocs/index.html>

위의 링크를 확인하시면 MBean interface 를 확인할 수 있습니다. MBean 객체도 결국 Java 이기  
때문에 많이 보셨던 Java API 와 동일하게 생각하시면 됩니다.

## MBean Type List

[AB\\_AlarmManager](#)  
[AdminAgent](#)  
[AdminOperations](#)  
[Adviso](#)  
[AdvisorNotificationMBean](#)  
[AntAgent](#)  
[AppManagement](#)  
[Application](#)  
[ApplicationManager](#)  
[AuthorizationGroupManager](#)  
[BulletinBoard](#)  
[CellSync](#)  
[Cluster](#)  
[ClusterBalance](#)  
[ClusterInfluence](#)  
[ClusterMgr](#)  
[CommandAssistance](#)  
[CommandRunner](#)  
[CompositionUnitManager](#)  
[ConfigRepository](#)  
[ConfigService](#)  
[ConnectionFactory](#)

## Server MBean

### All Parent MBeans:

[Stateful](#), [EventProvider](#), [J2EEManagedObject](#)

### Partial ObjectName:

WebSphere:\*, type=Server, j2eeType=J2EEServer

### MBean Server

Managed object for overall server process.

### Attribute Summary

java.lang.String	<a href="#">name</a>	The name of the server.
java.lang.String	<a href="#">shortName</a>	The short name of the server.
int	<a href="#">threadMonitorInterval</a>	Specifies the frequency (in seconds) at which threads will determine if they are hung.
int	<a href="#">threadMonitorThreshold</a>	Specifies the latency (in seconds) after which a thread is

Java API 하고 조금 다른 것은 우선 Partial ObjectName 이 있다는 것입니다. Partial ObjectName  
은 AdminControl.completeObjectName 을 이용해서 해당 Mbean 객체의 이름을 받아오기 위해서  
사용할 수 있습니다. 하단처럼, 위의 Server Mbean 의 Partial ObjectName 을 이용해서 Server 객  
체의 이름을 받아올 수 있습니다. (출력을 보시면 아시겠지만 Partial ObjectName 은 결국에는 객  
체 이름의 일부분 입니다.)

```
wsadmin>objNameString = AdminControl.completeObjectName('WebSphere:*,type=Server,j2eeType=J2EEServer')
wsadmin>print objNameString
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,version=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=UnManagedProcess
wsadmin>
```

또한, 해당 Interface 를 확인해보면 해당 객체에서 호출할 수 있는 메소드를 하단처럼 확인할 수 있습니다.

void	<a href="#"><u>stop()</u></a> Stop the server process.
void	<a href="#"><u>stopImmediate()</u></a> Stop the server process without going through application shutdown
void	<a href="#"><u>stop</u></a> (java.lang.Boolean callback, java.lang.Integer port) Stop the server process and callback.
void	<a href="#"><u>stop</u></a> (java.lang.String host, java.lang.Integer port) Stop the server process and callback to a host and port.
void	<a href="#"><u>restart()</u></a> Restart the server process.

해당 객체의 메소드들을 바로 invoke 명령을 이용해서 실행할 수 있으며 그렇기 때문에 무언가를 작동한다는 말을 하였던 것이었습니다. 이전에 했던 예제와 동일한 기능을 하는(WAS 서버 중지) invoke 를 해보도록 하겠습니다.

```
wsadmin>objNameString = AdminControl.completeObjectName('WebSphere:*,type=Server,j2eeType=J2EEServer')
wsadmin>print AdminControl.getAttribute(objNameString, 'state')
STARTED
wsadmin>AdminControl.invoke(objNameString, 'stop')
.
wsadmin>print AdminControl.getAttribute(objNameString, 'state')
WASX7015E: 명령 실행 중 예외: "AdminControl.getAttribute(objNameString, 'state')". 예외 정보:
com.ibm.websphere.management.exception.ConnectorNotAvailableException
org.apache.soap.SOAPException: [SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket:
java.net.ConnectException: Connection refused: connect; targetException=java.lang.IllegalArgumentException:
Exception: Error opening socket: java.net.ConnectException: Connection refused: connect]
wsadmin>
```

WAS 서버 중지 전에는 print AdminControl.getAttribute(objNameString, 'state') 를 통해서 'STARTED' 상태를 확인할 수 있으나 invoke 명령을 이용해서 stop 을 한 후에는 print AdminControl.getAttribute(objNameString, 'state') 명령 중에 connection refused 예외를 받습니다. 당연히, 해당 서버가 죽었기 때문에 본 예외를 받는 것입니다.

여기서 중요한 것은 Mbean Interface 를 인터넷을 통해서 확인 가능하며 그곳에 표현된 Partial ObjectName 을 이용해서 Mbean 객체의 이름을 받아올 수 있습니다. 뿐만 아니라, 그렇게 받아온 객체에다가 AdminControl.getAttribute 를 이용해서 멤버필드의 값을 받아올 수 있으며 AdminControl.invoke 명령을 이용해서 해당 Mbean Interface 에 설명된 메소드를 호출할 수 있다는 것입니다. 따라서 MBean Interface 를 잘 읽으실 줄만 알면 wsadmin 을 통해서 WAS 내의 모든 MBean 객체를 적절하게 조절가능하다는 점입니다.

사실 여기서 말씀드린 Mbean interface 를 보는 방법과 invoke 를 사용하는 방법과 잘 이해하셔도 wsadmin 을 맘껏 활용할 수 있습니다.

AdminConfig 객체는 WebSphere Application Server 컴포넌트를 작성하거나 수정하기 위해 구성 명령을 실행할 경우에 사용됩니다. 설명에서도 알 수 있듯이 관리콘솔을 이용해서 WAS 서버를 만들거나 JDBC Datasource 설정을 만드는 등 실질적인 WAS config 에 대한 작업을 할 수 해주는 객체입니다.

# attributes	# installResourceAdapter
# checkin	# list
# convertToCluster	# listTemplates
# create	# modify
# createClusterMember	# parents
# createDocument	# queryChanges
# createUsingTemplate	# remove
# defaults	# required
# deleteDocument	# reset
# existsDocument	# save
# extract	# setCrossDocumentValidationEnabled
# getCrossDocumentValidationEnabled	# setSaveMode
# getid	# setValidationLevel
# getObjectname	# show
# getSaveMode	# showall
# getValidationLevel	# showAttribute
# getValidationSeverityResult	# types
# hasChanges	# uninstallResourceAdapter
	# validate

여기서 자주 사용되는 명령은 attributes, getid, getObjectname, list, create, modify, save 명령입니다.

먼저 attributes 명령은 해당 type 의 객체의 모든 속성 리스트를 출력해 줍니다. 이러한 리스트를 통해서 어떠한 속성이 있는지 확인 가능합니다. (참고로 type 은 print AdminConfig.types() 를 이용해서 확인 가능합니다.)

```
wsadmin> print AdminConfig.attributes("Server")
adjustPort Boolean
changeGroupAfterStartup String
changeUserAfterStartup String
clusterName String
components Component(SecurityServer, SystemMessageServer, NodeAgent, EJBContainer, ExternallyManaged
HTTPServer, JobManager, WorkloadManagementServer, JMSServer, ApplicationServer, ApplicationContainer
, WebServer, SecurityServer, NameServer, SIPContainer, Agent, ServerComponent, WebContainer, CellMan
ager, Proxy, OnDemandRouter, ProxyServer, PortletContainer)+
customServices CustomService+
developmentMode boolean
errorStreamRedirect StreamRedirect
modelId String
name String
outputStreamRedirect StreamRedirect
parallelStartEnabled boolean
processDefinition ProcessDef(JavaProcessDef)
processDefinitions ProcessDef(JavaProcessDef)+
provisionComponents boolean
serverInstance ServerInstance
serverType String
services Service(DiagnosticProviderService, SIBService, TraceService, TPVService, ApplicationManagem
entService, FileTransferService, ConfigSynchronizationService, ApplicationProfileService, WorkManag
erService, DebugService, SchedulerService, EventInfrastructureService, TransportChannelService, WSByt
eBufferService, TransactionService, DynamicCache, II8NService, ObjectRequestBroker, MessageListenerS
ervice, CustomService, SessionManager, PMIService, ThreadPooManager, RASLoggingService, ActivitySe
sionService, JavaPersistenceAPIService, CompensationService, HAManagerService, AdminService, PlugInC
onfigService, ObjectPoolService, HTTPAccessLoggingService, CoreGroupBridgeService, StartupBeansSer
vice, CacheInstanceService, WorkAreaService, WorkAreaPartitionService)+
shortName String
stateManagement StateManagesable
statisticsProvider StatisticsProvider
uniqueId String
wsadmin>
```

Getid 는 해당 객체의 id 를 받아올 수 있는 명령입니다. 이렇게 받아온 id 를 통해서도 해당 객체들을 제어할 수 있습니다.

```
wsadmin>server = AdminConfig.getId('/Node:T400B92090901Node03/Server:server1/')
wsadmin>print server
server1(cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1|server.xml#Server_1183122130078)
wsadmin>
```

getObjectname 은 객체의 이름을 받아오는데 많이 보신 것 지 않으신가요? 바로 위에서 사용한 completeObjectName 과 동일하게 객체 이름을 받아오는 명령입니다.

```
wsadmin>print AdminConfig.getObjectname(server)
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,
ersion=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/
ervers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=Un
anagedProcess
wsadmin>
```

다음으로는 AdminConfig 에서 제일 많이 사용되는 명령중의 하나인 list 입니다. 말그대로 설정들의 list 를 가지고 오는 것을 의미하며 이 list 통해서도 id 를 확인할 수 있습니다.

```
wsadmin>print AdminConfig.list('Server')
server1(cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/servers/server1|server.xml#Server_1183122130078)
wsadmin>
```

객체의 id 를 가지고 오기 때문에 하단처럼 활용하여 위의 getObjectname 예제와 동일한 결과를 얻어올 수도 있습니다.

```
wsadmin>server = AdminConfig.list('Server')
wsadmin>print AdminConfig.getObjectname(server)
WebSphere:name=server1,process=server1,platform=proxy,node=T400B92090901Node03,j2eeType=J2EEServer,
ersion=7.0.0.7,type=Server,mbeanIdentifier=cells/T400B92090901Node01Cell/nodes/T400B92090901Node03/
ervers/server1/server.xml#Server_1183122130078,cell=T400B92090901Node01Cell,spec=1.0,processType=Un
anagedProcess
wsadmin>
```

Create 와 remove, modify 는 말그대로 설정을 생성하거나 삭제, 수정하는 작업을 할 수 있습니다.

마지막으로 save 명령은 AdminConfig.save() 로 실행할 수 있으며 말 그대로 설정 변경을 저장하는 기능을 합니다. 관리콘솔에서 설정을 변경한 다음에 저장을 눌러야지만 실제적으로 변경된 설정이 저장되는 것처럼 wsadmin 도 설정을 변경한 경우에는 save 명령을 통해서 저장을 해야지만 실제 변경이 저장됩니다.

AdminApp 객체는 응용프로그램을 관리할 경우에 사용합니다. 즉, WAS 서버에 설치된 어플리케이션을 관리하는 용도로 사용됩니다.

<pre># deleteUserAndGroupEntries # edit # editInteractive # export # exportDDL # New or updated for this feature pack newfeat exportFile # getDeployStatus # help # install # installInteractive</pre>	<pre># isAppReady # list # listModules # options # publishWSDL # searchJNDIReferences # taskInfo # uninstall # update # updateAccessIDs # updateInteractive # view</pre>
--	--

여기서 많이 사용되는 명령은 관리콘솔에서 어플리케이션을 설치, 업데이트를 하지 않고 script를 통하여 하기 위한 install, update 명령입니다

AdminApp.install('location\_of\_ear.ear', ['-node nodeName -cell cellName -server serverName']) 와 같은 양식의 명령으로 어플리케이션 install 이 가능하며

```
$AdminApp install c:\test\WEARTest.ear {-MapModulesToServers {"EJB_APPLIB"
ApplicationLibrary.jar,META-INF/ejb-jar.xml WebSphere:cell=cellName,server=serverName} }}
```

마지막으로 AdminTask 객체는 관리 명령을 실행할 경우에 사용합니다.

<pre># createServerType # createTCPEndPoint # getTCPEndPoint # help</pre>	<pre># listSSLRepertoires # listTCPEndPoints # listTCPThreadPools # updateAppOnCluster</pre>
---	--

관리명령이란 서버타입을 생성하거나 TCPEndPoint 를 생성하는 등의 관리명령을 위해서 사용하는 객체이나 많이 사용되지 않으므로 이번 강좌에서는 updateAppOnCluster 명령만 언급하고 간단히 넘어가도록 하겠습니다.

updateAppOnCluster 명령은 어플리케이션 업데이트 명령을 클러스터 환경에서 동시에 진행하는 것이 아니라 클러스터의 노드 단위로 순차적으로 업데이트 및 동기화 작업을 진행하는 명령입니다. 이 경우 모든 클러스터의 멤버가 중지되지 않으므로 서비스를 유지하면서 어플리케이션 업데이트가 가능합니다. 당연히 클러스터 환경을 지원할 수 있는 WAS ND 환경에서만 사용가능 합니다

다. 사용 방법도 단순한데 하단처럼 어플리케이션 update 후에 해당 사항을 save 를 통해서 저장하고 updateAppOnCluster 명령을 수행하면 됩니다. (이 명령은 대상 클러스터 범위에 있는 노드의 수에 따라 기본 커넥터 제한시간보다 많은 시간이 소요될 수 있습니다. SOAP 커넥터를 사용할 때에는 profile\_root/properties 디렉토리의 soap.client.props 파일에서, RMI 커넥터를 사용할 때에는 sas.client.props 파일에서 적절한 제한시간 값을 설정하셔야 합니다.)

```
AdminApp.update('app1', 'file', '[-operation update -contents c:/apps/app1/my.xml  
-contenturi app1.jar/my.xml]')  
AdminConfig.save()  
AdminTask.updateAppOnCluster('[-ApplicationNames app1]')
```

### Part 3. Wsadmin 을 위한 Jython 샘플

그럼 지금까지 배운 내용을 토대로 간단하지만 유용하게 활용할 수 있는 Jython 스크립트 샘플을 한번 보여드리도록 하겠습니다. WAS 운영중에 문제상황이 발생할 수 있으며, 문제 분석을 위해서 Lab 은 Trace 를 요청하는 경우가 많습니다. 이 경우를 대비해서 운영중에 해당 서버에 바로 Trace 를 걸 수 있는 Jython 스크립트 샘플입니다.

(하단과 같은 shell 과 같은 방법을 이용해서 실행가능합니다

```
./wsadmin.sh -lang jython -f runTraceToServer.py 서버명 트레이스 -user wasadm -password wasadm)
```

```
import sys
servername = '/Server:' + sys.argv[0] + '/'
print servername

server = AdminConfig.getid(servername)
print server

#APPSERVERSPEC="com.ibm.ws.xd.comm.*=all:"
APPSERVERSPEC=sys.argv[1]

def getNodeNameForServer(serverId):
    return serverId.split("/")[3]

def getProcessNameForServer(serverId):
    t = serverId.split("/")[5]
    t = t.split(":")[0]
    return t.split("|")[0]

def setRuntimeTraceForServer(server, traceString):
    nodeName = getNodeNameForServer(server)
    name = getProcessNameForServer(server)
    mbean = AdminControl.queryNames("type=TraceService,node=" + nodeName + ",process=" +
name + ",*")
    if(len(mbean) > 0):
        print "Setting runtime trace for " + nodeName + "/" + name
        AdminControl.setAttribute(mbean, "traceSpecification", traceString)
    else:
        print "Server " + nodeName + "/" + name + " is offline so runtime trace was not set."

def setTraceForServer(server, traceString):
```



```

nodeName = getNodeNameForServer(server)
name = getProcessNameForServer(server)
traceService = AdminConfig.list("TraceService", server)
print "Setting persistent trace for " + nodeName + "/" + name
AdminConfig.modify(traceService, [{"startupTraceSpecification", traceString}])

stype = AdminConfig.showAttribute(server,"serverType")
if (stype == "APPLICATION_SERVER"):
#     setTraceForServer(server,APPSERVERSPEC)
    setRuntimeTraceForServer(server,APPSERVERSPEC)

AdminConfig.save()

```

지금까지 wsadmin 에 대한 기본적인 이해와 Jython 이라는 script 언어를 활용하는 방법에 대한 강좌를 진행하였습니다. 처음에 언급한 것처럼 너무 방대한 양이기 때문에 필요한 것만 간단히 언급하였지만 부족한 부분이 많을 수도 있습니다. Wsadmin 은 반드시 사용해야 하는 툴은 아니지만 대용량 WAS 서버를 관리하는 경우나 반복 작업을 많이하는 경우에는 script 를 만들어 두고 자동화하는 것이 관리포인트를 줄일 수 있는 아주 좋은 방법이니 Jython 과 wsadmin 을 활용하는 방법을 이해하시고 실제로 운영에 활용해 보면 좋을 듯 합니다. 그럼 이번 강좌도 여기서 마무리 하도록 하겠습니다. 이만~~~~~휘리릭... ^^&

참고 1) IBM WebSphere Application Server v7.0 InfoCenter

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multipiplatform.doc/info/welcome\\_nd.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multipiplatform.doc/info/welcome_nd.html)

참고 2) IBM WebSphere Application Server v7.0 InfoCenter

- Getting started with wsadmin scripting

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/txml\\_script.html?resultof=%22%77%73%61%64%6d%69%6e%22%20](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/txml_script.html?resultof=%22%77%73%61%64%6d%69%6e%22%20)

참고 3) IBM WebSphere Application Server v7.0 InfoCenter

- MBean Interface

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.javadoc.doc/web/mbeanDocs/index.html>

※이 자료의 저작권은 작성자에게 있으며 유포는 자유로이 허용되나 상업적으로 이용은 금합니다.