

Anyframe Util System Plugin



Version 1.0.1

저작권 © 2007-2011 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

I. Introduction	1
II. System Utility	2
1. Architecture	3
1.1. SigarAccessor Architecture	3
1.2. File Monitor Architecture	4
1.3. Zip Architecture	4
1.4. Shell Script Architecture	4
2. SigarAccessor	6
2.1. System 정보 gathering	6
2.2. directRun 을 사용한 Sigar 접근	7
3. Utility	9
3.1. SystemInfo Utility	9
3.2. Network Utility	10
3.3. FileSystem Utility	10
3.4. FileMonitor	13
3.5. Zip Utility	14
4. Resources	17

I.Introduction

Util System Plugin은 공통적으로 활용할 수 있는 Utility 성 시스템 편의 기능을 제공한다. OS별 시스템, 파일시스템, 프로세스 등의 정보를 표준화된 API로 제공하는 오픈소스 Sigar 를 활용하여 System, Network, FileSystem 관련 정보들을 쉽게 얻을 수 있도록 오픈 소스 기능을 래핑하여 어플리케이션 내에서 utility 형태로 실행 가능토록 지원하고 있으며, SigarAccessor 라는 단일 통로를 통하여 다양한 시스템 리소스 정보를 gathering 하고 접근할 수 있도록 구조를 제공한다.

Util System Plugin은 오픈소스 Commons Compress 및 JDK 자체의 기능을 활용하는 자바 압축/해제 기능도 ZipUtil 을 통해 제공한다.

Shell Script 실행을 통한 시스템 기능 처리 또한 가능하며, Sigar 를 쓰지 않는 경우 java 환경에서 시스템 처리를 위해서는 Runtime 의 exec 메서드를 통해 OS 별 Shell Script (또는 단순 시스템 명령어) 를 실행하고 결과를 parsing 하여 원하는 정보를 추출하는 절차가 일반적으로 요구된다. 이러한 일련의 절차를 재사용 가능토록 모듈화하여 제공하는 ScriptExecutor 기능을 포함하고 있으나, 현재 제공되는 실행 기능은 디렉토리 size 조회, 메모리 사용량 조회, mac address, port scan 정보 조회로 제한적이다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 util-system plugin을 설치한다.

```
mvn anyframe:install -Dname=util-system
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

Dependent Plugins

Plugin Name	Version Range
core [http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.0.3/reference/htmlsingle/core.html]	2.0.0 > *

II. System Utility

Util System Plugin에서 제공하는 유틸리티는 아래와 같다.

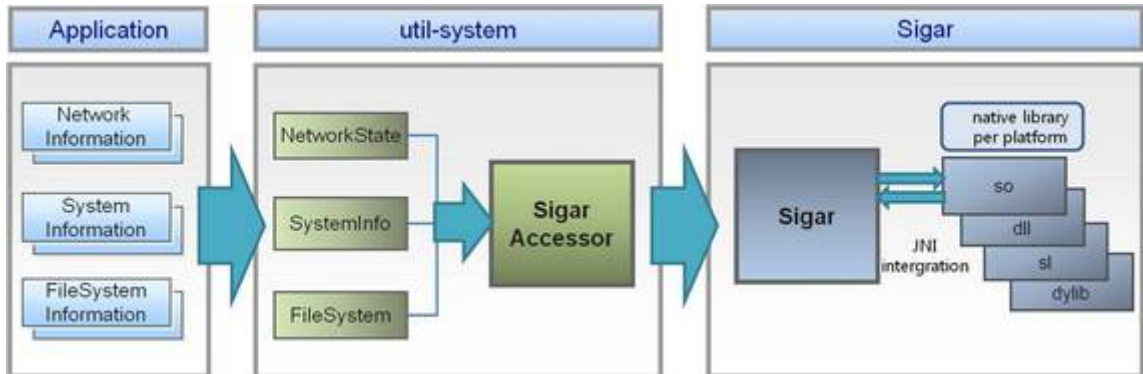
Name	Description
SystemInfoUtil	디스크 속성, 시스템 정보, 프로세스 정보, 메모리 정보, 클라이언트 정보등의 기능을 제공한다.
NetworkStateUtil	네트워크 상태 및 IP정보, Mac 주소정보, 포트스캔한 정보 기능을 제공한다.
FileSystemUtil	파일 및 디렉토리 속성 정보, 권한 체크, 복사,삭제,생성 , 압축 등의 기능을 제공한다.
FileMonitor	디렉토리 감시 기능을 제공한다. 감시 대상 디렉토리내에 추가/수정/삭제된 파일에 대한 정보를 로그로 기록할 수 있다.
ZipUtil	파일 및 디렉토리 압축/해제 기능을 제공한다.

1.Architecture

1.1.SigarAccessor Architecture

Anyframe 에서는 JNI 라이브러리를 사용하는 SIGAR 의 초기 구동 환경 설정을 간편화하기 위한 SigarAccessor 를 제공한다.

다음 그림은 Sigar 를 사용하기 위한 Anyframe 시스템 Architecture 다.



SIGAR (System Information Gatherer and Reporter) 는 다양한 platform/language 를 지원하는 오픈 소스(Apache License, Version 2.0) 라이브러리로 Java, Perl, .Net 등에서 OS 와 하드웨어 수준의 정보에 접근하기하기 위한 command-line 툴이다.

SIGAR 는 System Management/Monitoring Tool Suite 인 Hyperic HQ 의 핵심 서브 프로젝트로 처음에는 Java 플랫폼에서 low-level의 하드웨어/OS 정보에 대한 유연한 접근 및 측정이 어려운 문제를 극복하기 위해 만들어 졌다. 현재 Hyperic 은 SpringSource 에 인수되었으며 라이선스 정책도 최근 Apache License 로 변경되어 사용에 제약이 없어지게 되었다. JNI(Java Native Interface) 기반으로 플랫폼에 따라 상이한 시스템 처리도 Native 라이브러리를 사용하여 빠르고 강력한 low-level 처리로 수행하며 Java JNI Binding 을 통하여 유연하게 Java 환경에 통합된다.

SIGAR 에서 제공하는 주요 기능은 다음과 같다.

- 시스템 메모리, SWAP, cpu, load average, uptime..
- 프로세스당 메모리정보, 자격인증정보, 상태, arguments, environment
- 파일시스템 정보
- 네트워크 정보 및 설정
- TCP 와 UDP 연결 테이블 정보
- 네트워크 라우트 테이블 정보

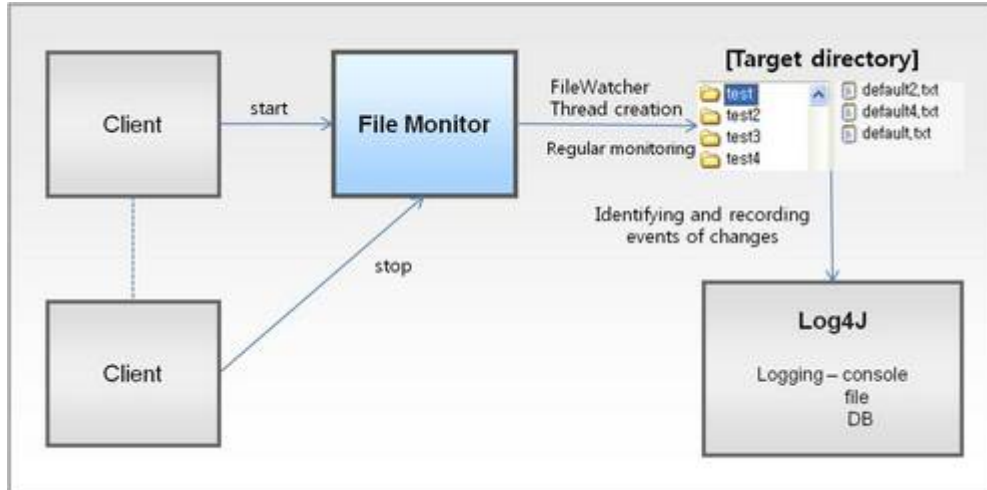
이런 다양한 기능의 Sigar 의 기능을 쉽게 활용하기 위해 Anyframe 에서는 SigarAccessor 를 제공해 시스템, 네트워크, 파일에 관한 정보 제공을 손쉽게 한다. SigarAccessor 의 기능은 아래와 같다.

- Sigar 구동환경 초기화
- 기본 System resource gathering 및 초기화
- 실시간 변경 resource 갱신기능

1.2. File Monitor Architecture

Anyframe 에서는 FileMonitor 라는 utility 를 통해 Sigar 에서 제공하는 FileWatcherThread 를 생성시켜 주기적으로 디렉토리 내 파일의 변경 이벤트를 감지할 수 있는 기능을 제공한다. 감시 대상 디렉토리 내에 추가/수정/삭제된 파일에 대한 정보는 log4j를 활용하여 로그로 기록한다.(Appender 지정에 따라 console, file, DB 등 다양한 target 설정 가능)

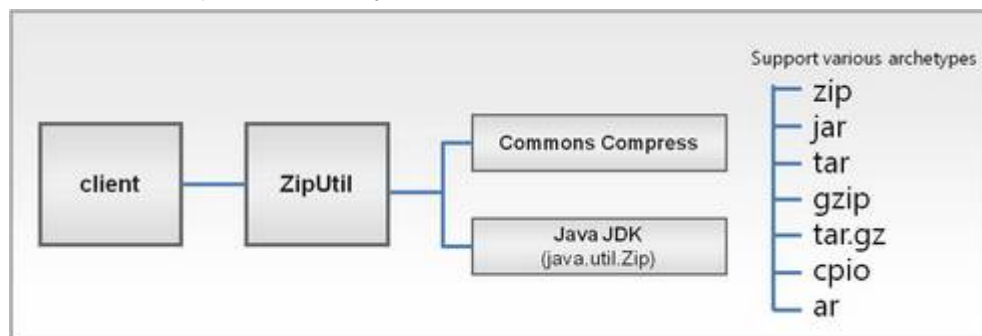
다음 그림은 FileMonitor 를 위한 Anyframe 시스템 Architecture 이다.



1.3. Zip Architecture

Anyframe 에서는 Apache Commons Compress 기능을 활용하여 zip, tar, tar.gz 등의 파일 압축/해제를 지원한다.

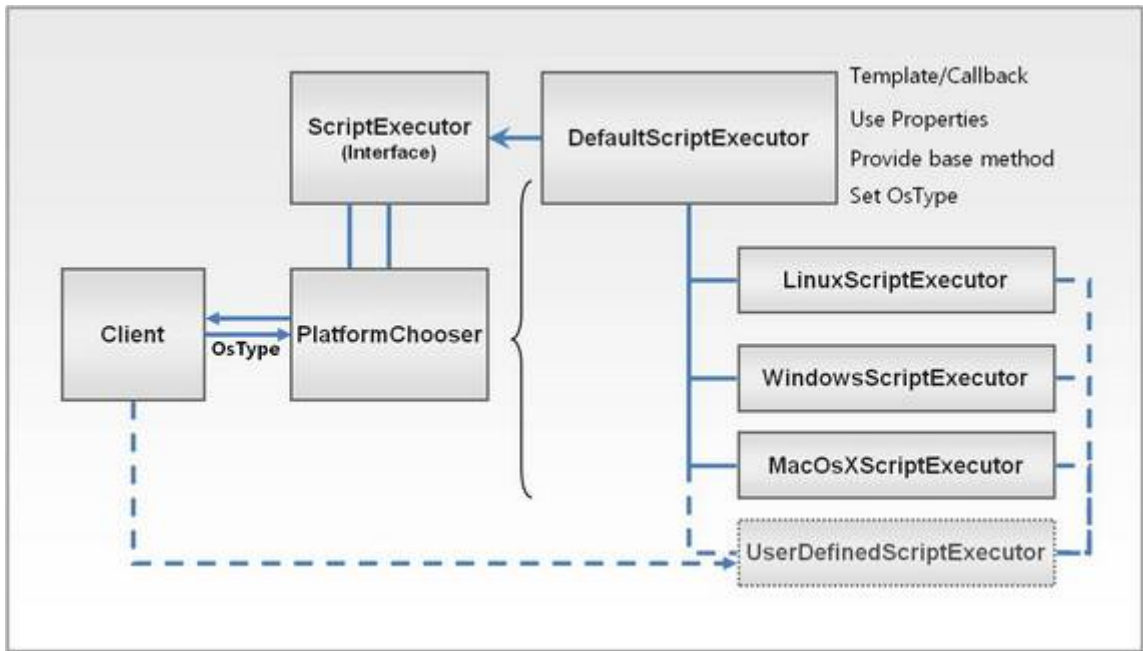
다음 그림은 파일 압축/해제를 위한 Anyframe 시스템 Architecture 이다.



1.4. Shell Script Architecture

Anyframe 에서는 자바 환경에서 Shell Script 실행 및 결과 처리를 유연하게 지원하기 위한 기능을 제공한다.

다음 그림은 Shell Script 실행을 위한 Anyframe 시스템 Architecture 이다.



2.SigarAccessor

Sigar 에서 제공하는 JNI 연동 기능을 사용하기 위해서는 Sigar Library Path 를 잡아줘야 한다. SigarAccessor 에서는 기본으로 JVM working directory 기준으로 sigar-bin/lib 폴더에 대해서 library path 를 잡아주고 있다. 따라서 기본 경로가 바라보는 위치에 플러그인 샘플에 포함하고 있는 JNI native library 묶음인 sigar-bin/lib 를 copy 해 놓으면 기동에 문제가 없다. (기본 working directory 는 WAS 별/build Tool 별로 상이하다.) Sigar Library Path 는 JVM 기동 옵션이나 플랫폼 별 library path 설정을 통해 특정 프로젝트에 맞게 변경 가능하다. (Windows 인 경우 path 만 잡아도 가능, Linux/Solaris 인 경우 LD_LIBRARY_PATH, AIX 인 경우 LIBPATH 설정 등, java.library.path 시스템 변수에 직접 설정/append 하는 것도 가능) Spring Container 기동 time 에 System.setProperty("java.library.path", 기존path + sigarLibPath) 형태로 재설정 하는 것도 가능하며 관련 샘플은 util-system-demo plugin의 UtilSystemController 의 @PostConstruct 로 지정한 initSigar() 메서드를 참조하라.

2.1.System 정보 gathering

SigarAccessor 에서는 기본적으로 system 관련 정보들을 수집해서 가지고 있다. 내부적으로 SysInfo 라는 VO 객체를 넘겨준다. 해당 정보를 이용하고자 하면 SysInfo 안에 있는 아래 명시된 VO 객체를 꺼내서 사용하면 된다.

정보 수집 Method	Description	VO	실시간 정보수집 필요여부
gatherVersionInfo	SigarInfo (OS, Java 버전 정보 수집)	NativeInfo, OsInfo, JavalInfo	No
gatherUptimeInfo	UptimeInfo (정보 - 시스템 부팅 시간 및 현재까지 실행 시간)	UptimeInfo	Yes
gatherCpuInfo	CpuInfo (정보 - 벤더, 모델, Mhz, CPU 갯수, Core 갯수, Core 당 소켓수...등)	CpuInfo	No
gatherMemoryInfo	MemoryInfo (메모리 정보-전체메모리, 할당된메모리, 유휴메모리...등)	MemoryInfo	Yes
gatherFileSystemInfo	SystemInfo (기본 정보-전체파일시스템에 대한 디렉토리명, 파일명...등)	FileSystemInfo	No
gatherFileSystemUsageMap	SystemUsageMap (기본 정보-전체 디스크 크기, 사용중인 디스크크기, 사용가능한 디스크 크기...등)	FileSystemUsageInfo	Yes
gatherProcessStatsInfo	ProcessStatsInfo (Process stats 정보)	ProcessStatInfo	Yes
gatherProcessInfo	ProcessInfo (전체 정보 - CPU 정보, 실행정보, 상태정보, 시간 정보...등)	ProcessInfo ProcessCpuInfo, ProcessExeInfo, ProcessStateInfo, ProcessMemoryInfo	Yes
gatherNetworkInfo	NetworkInfo (네트워크 정보 - ip, mac, network mask...등)	NetworkInfo	No
gatherNetworkInfo	NetworkInfo (네트워크 정보 - ip, mac, network mask...등)	NetworkInfo	No

정보 수집 Method	Description	VO	실시간 정보수집 필요여부
gatherUlimitInfo	정보 - 데이터, 파일, 시간, 코어덤프, 메모리...등	UlimitInfo	Yes

2.2.directRun 을 사용한 Sigar 접근

시스템 구동 시 설정한 SigarAccessor 의 기본 Shell 을 사용하여 Sigar API의 registered Commands 로 등록된 명령어를 바로 실행하여 결과를 console 에서 확인할 수 있다.

registeredCmd 의 예 :
df,du,ls,iostat,free,pargs,penv,pfile,pmodules,pinfo,cpuinfo,ifconfig,uptime,ps,pidof,kill,netstat,netinfo,nfsstat,route,version,mps,sysinfo,time,ulimit,who, [Windows 인 경우 service,fversion]

```

SigarAccessor.directRun("df", new String[] {});
SigarAccessor.directRun("du", new String[] { "." });
SigarAccessor.directRun("ls", new String[] { "." });
SigarAccessor.directRun("iostat", new String[] {});
SigarAccessor.directRun("free", new String[] {});
SigarAccessor.directRun("pargs", new String[] {});
SigarAccessor.directRun("penv", new String[] {});
SigarAccessor.directRun("pfile", new String[] {});
SigarAccessor.directRun("pmodules", new String[] {});
SigarAccessor.directRun("pinfo", new String[] {});
SigarAccessor.directRun("cpuinfo", new String[] {});
SigarAccessor.directRun("ifconfig", new String[] {});
SigarAccessor.directRun("uptime", new String[] {});
SigarAccessor.directRun("ps", new String[] {});
String topPid = sigarCmdTest.getTopPid();

SigarAccessor.directRun("pidof", new String[] { topPid });
SigarAccessor.directRun("kill", new String[] { topPid });

SigarAccessor.directRun("netstat", new String[] {});
SigarAccessor.directRun("netinfo", new String[] {});
SigarAccessor.directRun("nfsstat", new String[] {});
SigarAccessor.directRun("route", new String[] {});
SigarAccessor.directRun("version", new String[] {});

// PTQL
SigarAccessor.directRun("mps", new String[] { "State.Name.eq=java" });

SigarAccessor.directRun("sysinfo", new String[] {});
SigarAccessor.directRun("time", new String[] { "ps" });
SigarAccessor.directRun("ulimit", new String[] {});
SigarAccessor.directRun("who", new String[] {});
if (SigarLoader.IS_WIN32) {
    SigarAccessor.directRun("service", new String[] { "Alerter", "state" });
    SigarAccessor.directRun("fversion", new String[] { "C:\\WINDOWS\\winhelp.exe" });
}

```

df 를 실행한 결과는 아래와 같다.

Filesystem	Size	Used	Avail	Use%	Mounted on	Type
C:\	117G	30G	87G	26%	C:\	NTFS/local
D:\	349G	40G	308G	12%	D:\	NTFS/local

```
E:\          0  0  0  -  E:\          cdrom/cdrom
```

3.Utility

Anyframe에서는 system, network, file 관련 정보들을 손쉽게 사용하기 위해 관련 유틸리티들을 제공한다.

3.1.SystemInfo Utility

SystemInfoUtil 클래스를 통해 시스템 정보를 얻을 수 있다.

Method	Description	Return
getDiskProperty()	파일 시스템 목록 조회 - 디렉토리명, 파일명 등	List<FileSystemInfo>
getDiskProperty (final String name)	특정한 파일 시스템 조회 - 디렉토리명, 파일명 등	FileSystemInfo
getDiskCapacity()	전체파일시스템 유효정보 조회 - 전체메모리사이즈,사용중인사이즈,퍼센트,사용가능한 메모리사이즈등	Map<String, FileSystemUsageInfo>
getDiskCapacity (final String name)	특정한 파일시스템 유효정보 조회 - 전체메모리사이즈,사용중인사이즈,퍼센트,사용가능한 메모리사이즈등	FileSystemUsageInfo
getMountedDiskCapacity	마운트된 특정한 파일 시스템 유효정보 조회 - 전체메모리사이즈,사용중인사이즈,퍼센트,사용가능한 메모리사이즈	FileSystemUsageInfo
replacePath (final String path)	OS 에 맞는 드라이브 경로 문자로 변경	String
getSystemInfo()	OS 정보	OsInfo
getProcessStat()	시스템 프로세스 status 정보 - 프로세스 정보 . 전체갯수, 실행중인갯수, 상태별 프로세스 갯수, 스레드갯수 등	ProcessStatInfo
getProcessList (final String[] args)	프로세스 실행 정보 조회	Map<Long, ProcessInfo>
getPs()	ps 를 실행한 결과와 같은 정보 - 현재 실행되는 프로세스의 실행 상태	List<String>
getWho()	who 를 실행한 결과와 같은 정보 - 현재 시스템에 접속한 사용자 확인	List<String>
getMemoryCapacity (final MemRegion memRegion)	memRegion 입력 값에 따라 전체메모리사이즈,사용중인사이즈,사용가능한 메모리사이즈 중에 하나를 보여준다.	long
getLs (final String name)	ls 를 실행한 결과와 같은 정보 - 해당 디렉토리 및 파일에 대한 ls 정보	String
getLsList (final String name)	해당하는 디렉토리 하위의 파일들에 대한 ls 목록 조회	List<String>
getCpuInfo()	해당 시스템의 Cpu 정보 조회	CpuInfo
getClientIP (final HttpServletRequest request)	클라이언트의 IP 주소 정보 조회	String
getClientOS (final HttpServletRequest request)	클라이언트의 OS 정보 조회	String
getClientBrowser (final HttpServletRequest request)	클라이언트의 브라우저와 버전 정보	String

3.2. Network Utility

NetworkStateUtil 클래스를 통해 네트워크 상태 및 IP, Mac Address 등에 대한 정보를 얻을 수 있다.

Method	Description	Return
testPing (final String destHost, final int timeout)	네트워크 상태를 확인한다.	boolean
getNetworkInfoList ()	전체 네트워크 목록에 대한 ip, mac, network mask 등 조회	Map<String, NetworkInfo>
getMyIPInfo ()	NetInfo 를 실행한 결과와 같은 형태의 정보 조회	NetworkInfo
getMyIPList ()	전체 네트워크 목록에 대한 ip 목록 조회	List<String>
getMyMacAddressList ()	전체 네트워크 목록에 대한 Mac Address 목록	List<String>
getPortScan ()	netstat 를 실행한 결과와 같은 형태의 정보 조회. 명령어 옵션으로 "a", "p" 사용	List<String>
getPortScan (final String[] args)	netstat 를 실행한 결과와 같은 형태의 정보 조회. Siga 에서 지원하는 Netstat 옵션 ("l", "a", "n", "p", "s", "t", "u", "w", "x")	List<String>
getRoute ()	route 를 실행한 결과와 같은 형태의 정보 조회.	List<String>

3.3. File System Utility

FileSystemUtil 클래스를 통해 파일 및 디렉토리 속성 정보, 권한 체크, 복사, 삭제, 생성 등의 기능을 사용할 수 있다.

Method	Description	Return
canRead (String file)	대상 파일 또는 디렉토리를 읽을 수 있는지 여부 체크	boolean
canWrite (String file)	대상 파일 또는 디렉토리를 쓸 수 있는지 여부 체크	boolean
copyDirectory (final String srcDir, final String destDir)	원본디렉토리를 대상 디렉토리명으로 복사한다.	void
copyDirectory (final String srcDir, final String destDir, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. preserveFileDate가 true면 원본디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final FileDir filedir)	원본디렉토리를 대상 디렉토리명으로 복사한다. fileDir가 'file'이면 파일만 'directory'면 디렉토리만 복사한다.	void
copyDirectory (final String srcDir, final String destDir, final FileDir filedir, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. fileDir가 'file'이면 파일만 'directory'면 디렉토리만 복사한다. preserveFileDate가 true면 원본디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final String extension)	원본디렉토리를 대상 디렉토리명으로 복사한다. 입력된 파일 확장자랑 일치하는 파일들만 복사한다.	void

Method	Description	Return
copyDirectory (final String srcDir, final String destDir, final String extension, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. 입력된 파일 확장자랑 일치하는 파일들만 복사한다. preserveFileDate가 true면 원본 디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final Date cutoffDate, final boolean acceptOlder)	원본디렉토리를 대상 디렉토리명으로 복사한다. 입력된 파일 확장자랑 일치하는 파일들만 복사한다. 입력받은 날짜를 기준으로 최종 수정일이 입력받은 날짜보다 적거나 많은 파일만 복사함.	void
copyDirectory (final String srcDir, final String destDir, final Date cutoffDate, final boolean acceptOlder, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. 입력된 파일 확장자랑 일치하는 파일들만 복사한다. 입력받은 날짜를 기준으로 최종 수정일이 입력받은 날짜보다 적거나 많은 파일만 복사함. preserveFileDate가 true면 원본 디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final long threshold, final boolean acceptLarger)	원본디렉토리를 대상 디렉토리명으로 복사한다. 대상 디렉토리에 대해 입력받은 사이즈를 기준으로 파일 사이즈가 입력받은 사이즈보다 적거나 많은 파일 복사	void
copyDirectory (final String srcDir, final String destDir, final long threshold, final boolean acceptLarger, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. 대상 디렉토리에 대해 입력받은 사이즈를 기준으로 파일 사이즈가 입력받은 사이즈보다 적거나 많은 파일 복사 preserveFileDate가 true면 원본디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final long minSize, final long maxSize)	원본디렉토리를 대상 디렉토리명으로 복사한다. 대상 디렉토리에 대해 입력받은 최대, 최소 사이의 사이즈의 파일 목록 복사	void
copyDirectory (final String srcDir, final String destDir, final long minSize, final long maxSize, final boolean preserveFileDate)	원본디렉토리를 대상 디렉토리명으로 복사한다. 대상 디렉토리에 대해 입력받은 최대, 최소 사이의 사이즈의 파일 목록 복사 preserveFileDate가 true면 원본디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectory (final String srcDir, final String destDir, final String name, final FileNameSearch fileNameSearch)	파일명의 검색 조건에 따라 원본디렉토리를 대상 디렉토리명으로 복사한다. 검색하고자 하는 파일 명 또는 prefix , suffix	void
copyDirectory (final String srcDir, final String destDir, final String name, final FileNameSearch fileNameSearch, final boolean preserveFileDate)	파일명의 검색 조건에 따라 원본디렉토리를 대상 디렉토리명으로 복사한다. 검색하고자 하는 파일 명 또는 prefix , suffix preserveFileDate가 true면 원본디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyDirectoryToDirectory (final String srcDir, final String destDir)	원본 디렉토리를 대상 디렉토리의 하위로 복사한다.	void

Method	Description	Return
moveDirectory (final String srcDir, final String destDir)	디렉토리 이동-원본 디렉토리가 대상 디렉토리 명으로 이동한다.	void
moveDirectoryToDirectory (final String src, final String destDir, final boolean createDestDir)	디렉토리 이동-원본 디렉토리가 대상 디렉토리 하위로 이동한다.	void
makeDirectory (final String directory)	디렉토리 생성	void
deleteFileDirectory (final String src)	해당 디렉토리 및 파일을 삭제한다.	void
copyFile (final String srcFile, final String destFile)	파일 복사	void
copyFile (final String srcFile, final String destFile, final boolean preserveFileDate)	파일 복사. preserveFileDate가 true면 원본 디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
copyFileToDirectory (final String srcFile, final String destDir)	파일을 대상 디렉토리에 복사	void
copyFileToDirectory (final String srcFile, final String destDir, final boolean preserveFileDate)	파일을 대상 디렉토리에 복사. preserveFileDate가 true면 원본 디렉토리의 파일 최종 수정일을 그대로 사용하고 false면 현재 일자를 최종 수정일로 설정한다.	void
moveFile (final String srcFile, final String destFile)	파일 이동	void
moveFileToDirectory (final String srcFile, final String destDir, final boolean createDestDir)	파일을 대상 디렉토리 하위로 이동	void
getInformation (final String name)	해당하는 파일이나 디렉토리의 속성정보 반환(사이즈, 권한, 타입, etc)	FileInfo
getFileList (final String directory, final String[] extensions, final boolean recursive)	해당하는 디렉토리 밑의 파일 목록 조회	File[]
getFileList (final String directory, final Date date, final boolean acceptOlder)	대상 디렉토리에 대해 입력받은 날짜를 기준으로 최종 수정일이 입력받은 날짜보다 적거나 많은 파일 목록 조회	File[]
getFileList (final String directory, final long size, final boolean acceptLarger)	대상 디렉토리에 대해 입력받은 사이즈를 기준으로 파일 사이즈가 입력받은 사이즈보다 적거나 많은 파일 목록 조회	File[]
getFileList (final String directory, final long minSizeInclusive, final long maxSizeInclusive)	대상 디렉토리에 대해 입력받은 최대, 최소 사이의 사이즈의 파일 목록 조회	File[]
getFileList (final String directory, final String name, final FileNameSearch fileNameSearch)	파일명에 따른 대상 디렉토리의 파일 검색. 검색하고자 하는 파일 명 또는 prefix , suffix	File[]
existsFile (final String filepath)	대상 파일이 존재하는지 체크	boolean
existsDir (final String dirpath)	대상 디렉토리가 존재하는지 체크	boolean
existsDir (final String srcDir, final String fromDate, final String toDate)	해당 디렉토리의 마지막 수정일이 입력받은 날짜 사이에 존재하는지 체크	boolean

3.4.FileMonitor

FileMonitor 클래스를 통해 디렉토리 감시 기능을 사용할 수 있으며 감시 대상 디렉토리내에 추가/수정/삭제된 파일에 대한 정보를 로그로 기록할 수 있다. 디렉토리내 파일이 많은 경우 성능 저하가 발생할 수 있으며, Windows 인 경우 감시 대상 디렉토리 내에 한글 파일명이 존재하는 경우 문제가 발생하므로 유의한다.

- 파일에 기록하기 위한 log4j 설정의 예이다.

```
<appender name="file"
  class="org.apache.log4j.DailyRollingFileAppender">
  <param name="File"
    value="logs/fileMonitor.log" />
  <param name="Append" value="true" />
  <param name="DatePattern" value="'. 'yyyy-MM-dd" />
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern"
      value="%d %5p [%c] %m%n" />
  </layout>
</appender>
..
<logger name="fileMonitor" additivity="false">
  <level value="INFO"/>
  <appender-ref ref="file"/>
</logger>
```

- 실제 사용예는 아래와 같다.

```
Log log = LogFactory.getLog("fileMonitor");
FilewatcherThread filewatcherThread = FileMonitor.start(log, targetDir, 5000);
```

- 아래는 모니터링 주기 5초로 디렉토리 감시 start ~ stop 까지의 로그 파일 기록 예이다. 파일의 생성/변경/삭제 내용이 prefix(NEW/MOD/DEL) 및 파일명으로 나타나며 이때 변경시간(Mtime), 변경된 Size 등이 before|after 형식으로 나타남을 확인할 수 있다.

```
2011-01-19 10:13:01,625 INFO [anyframe.core.util.system.FileMonitor] FilewatcherThread
started.
2011-01-19 10:13:06,718 INFO [anyframe.core.util.system.FileMonitor] NEW$:
\workspace_neis_helios\local.common-component.system\test\default2.txt${Mtime: 1월 19
10:13}{Size: 52}
2011-01-19 10:13:11,750 INFO [anyframe.core.util.system.FileMonitor] NEW$:
\workspace_neis_helios\local.common-component.system\test\default3.txt${Mtime: 1월 19
10:13}{Size: 52}
2011-01-19 10:13:11,765 INFO [anyframe.core.util.system.FileMonitor] NEW$:
\workspace_neis_helios\local.common-component.system\test\default4.txt${Mtime: 1월 19
10:13}{Size: 16}
2011-01-19 10:13:11,765 INFO [anyframe.core.util.system.FileMonitor] NEW$:
\workspace_neis_helios\local.common-component.system\test\default5.txt${Mtime: 1월 19
10:13}{Size: 16}
2011-01-19 10:13:11,812 INFO [anyframe.core.util.system.FileMonitor] MOD$:
\workspace_neis_helios\local.common-component.system\test\default2.txt${Mtime: 1월 19
10:13|1월 19 10:13}{Size: 52|16}
2011-01-19 10:13:16,828 INFO [anyframe.core.util.system.FileMonitor] DEL$:
\workspace_neis_helios\local.common-component.system\test\default5.txt
2011-01-19 10:13:16,843 INFO [anyframe.core.util.system.FileMonitor] MOD$:
\workspace_neis_helios\local.common-component.system\test\default4.txt${Mtime: 1월 19
10:13|1월 19 10:13}{Size: 16|24}
2011-01-19 10:13:16,859 INFO [anyframe.core.util.system.FileMonitor] DEL$:
\workspace_neis_helios\local.common-component.system\test\default3.txt
```

```
2011-01-19 10:13:21,625 INFO [anyframe.core.util.system.FileMonitor] FileWatcherThread
stopped.
```

- FileMonitor 를 위해 singleton 방식의 startSingleton()/stopSingleton 과 멀티스레드 방식의 start()/stop 두가지 타입의 메소드를 지원한다.
- singleton 방식의 사용예는 아래와 같다.

```
String targetDir = "test";
init(targetDir);

//파일모니터 시작
FileMonitor.startSingleton(targetDir, 5000);

중략...

Thread.sleep(20 * 1000);

//파일모니터 종료
FileMonitor.stopSingleton();
```

- 멀티스레드 방식의 사용예는 아래와 같다.

```
String targetDir = "test4";
init(targetDir);

Log log = LogFactory.getLog("fileMonitor");
//파일모니터 시작
FileWatcherThread fileWatcherThread = FileMonitor.start(log, targetDir, 5000);

중략...

//파일모니터 종료
FileMonitor.stop(log, fileWatcherThread);
```

3.5.Zip Utility

ZipUtil 클래스를 통해 파일 및 디렉토리 압축/해제 기능을 제공한다. Java JDK 및 Apache Commons Compress 를 활용한다. Zip 파일의 경우는 추가로 Encoding 을 지정할 수 있으며, 특히 한글 파일명이 포함된 압축파일의 해제 시 원본 Encoding 에 유의하여 지정해 주어야 한다.

- 파일 및 디렉토리 압축

Method	Description
compressZip (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Zip 압축을 처리한다.
compressZip (String targetDir, String destArchiveStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Zip 압축을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성한다.
compressZip (String targetDir, String destArchiveStr, final String encoding)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Zip 압축을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성하며, 인자로 전달된 Encoding 을 적용하여 압축파일 내 Entry filename 이 작성된다.
compressJar (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Jar 압축(Zip 압축과 유사)을 처리한다.

Method	Description
compressJar (String targetDir, String destArchiveStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Jar 압축(Zip 압축과 유사)을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성한다.
compressJar (String targetDir, String destArchiveStr, final String encoding)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Jar 압축(Zip 압축과 유사)을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성하며, 인자로 전달된 Encoding 을 적용하여 압축파일 내 Entry filename 이 작성된다.
compressTar (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Tar 압축(하나의 파일로 묶어줌)을 처리한다.
compressTar (String targetDir, String destArchiveStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Tar 압축(하나의 파일로 묶어줌)을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성한다.
compressGzip (File targetArchiveOrFile, String destArchiveStr)	Gzip 압축을 처리한다. Gzip 압축 대상은 주로 tar 압축파일이며 compressTarGz 메서드에서 내부적으로 해당 메서드를 호출하게 된다.
compressTarGz (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 tar.gz 압축을 처리한다.
compressTarGz (final String targetDir, final String destTarGzStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 tar.gz 압축을 처리한다. 이때 압축파일을 인자로 전달받은 destTarGzStr 경로에 생성한다. 내부적으로 먼저 compressTar 압축 후 생성된 Tar 파일을 다시 compressGzip 메서드를 호출하여 Gzip 압축하게 된다. 최종 tar.gz 압축파일과 동일 위치에 tar 파일도 함께 생성됨에 유의한다.
compressAr (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Ar 압축을 처리한다.
compressAr (String targetDir, String destArchiveStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Ar 압축을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성한다. filename 길이 16 byte 제한이 있으므로 유의한다.
compressCpio (String targetDir)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Cpio 압축을 처리한다.
compressCpio (String targetDir, String destArchiveStr)	디렉토리 또는 파일에 대한 압축대상 경로를 인자로 받아 Cpio 압축을 처리한다. 이때 압축파일을 인자로 전달받은 destArchiveStr 경로에 생성한다.

- 파일 및 디렉토리 해제

Method	Description
decompressZip (String targetZipFileStr)	Zip 압축파일 경로를 인자로 해당 압축파일을 해제한다. 이때 해당 압축파일과 동일한 path 에 압축을 해제하게 된다.
decompressZip (String targetZipFileStr, String destDirStr)	Zip 압축파일을 해제한다. 이때 인자로 전달받은 대상경로에 압축을 해제하게 된다.
decompressZip (String targetZipFileStr, String destDirStr, final String encoding)	Zip 압축파일을 해제한다. 이때 인자로 대상경로와 Encoding 을 전달 받아 처리한다
decompressGzip (String gzipFile)	Gzip 파일을 압축해제하고 해당 파일을 돌려준다.

Method	Description
decompressArchive (String targetCompressFileStr)	Commons Compress 의 ArchiveStreamFactory 를 사용하여 압축파일 (Zip/Jar/Tar/Ar/Cpio)을 해제한다.
decompressArchive (String targetCompressFileStr, String destDirStr)	Commons Compress 의 ArchiveStreamFactory 를 사용하여 압축파일 (Zip/Jar/Tar/Ar/Cpio)을 해제한다. 이때 인자로 전달받은 대상경로에 압축을 해제하게 된다.
decompressJavaZip (String targetZipFileStr)	자바 기본 기능(java.util.zip)을 사용하여 Zip 압축파일을 해제한다. 압축파일 내의 filename 은 UTF-8 이어야 함에 유의한다.
decompressJavaZip (String targetZipFileStr, String destDirStr)	자바 기본 기능(java.util.zip)을 사용하여 Zip 압축파일을 해제한다. 압축파일 내의 filename 은 UTF-8 이어야 함에 유의한다. 이때 인자로 전달받은 대상경로에 압축을 해제하게 된다.

일반적으로 사용자가 local PC 에서 Windows 환경의 압축 프로그램으로 생성한 zip 압축 파일은 EUC-KR 로 인코딩되어 있는 경우가 많으므로 이런 경우 EUC-KR 을 지정하여 ZipUtil 의 decompressZip 메서드를 실행해야 한다. 서버 상의 Application 에서 Zip 압축을 생성하는 경우 UTF-8 인코딩을 지정하여 처리하는 것이 타 시스템 연동 유연성이 높으며 EUC-KR 로 decompressZip 을 실행하여도 에러가 발생하지 않으므로 참고한다.

4.Resources

- 참고자료
 - Sigar API [<http://support.hyperic.com/display/SIGAR/Home>]
 - Apache Commons Compress [<http://commons.apache.org/compress/>]