Using JAVA in daemons and Services.

http://jakarta.apache.org/commons/daemons/

jakarta-daemons

- History
- Why to use it?
- jsvc/procrun.
- Mixing java and native C programs.
- Supported JVM's.
- Futur.
- Comments.

History

The 26th of june 2001 Pier Paolo Fumagalli committed the first code in jakarta-tomcat-service/java.

During 2002 it moves to jakarta-commons-sandbox as daemon.

Mid June 2003 it gets promoted to jakarta-commons proper.

Since then a lot of features have been added to it.

Why to use it?

- To get a program written in Java to be started automatically when the box is start (rc files or services).
- To run Tomcat on port 80 as non-root user.
- To replace shell scripts.
- To get the JVM restarted in case of crash.
- To make sure that things like session informations are saved when stopping the box.
- To allow different Java programs to be started in a defined sequence.

jsvc/procrun

One for Unix like systems, one for win32.

Why: Because the code doesn't use (yet) APR... Well it is quite different to start,install service in win32 then to have shell script in /etc/rc.d/.

jsvc:

- jsvc

procrun:

- Prunsrv service application (console).
- Prunmgr monitor and configuration (gui).

jsvc parameters

-jvm <JVM name>
-cp / -classpath
-home <directory> path to JDK or JRE installation
-version
-help / -?
-nodetach
-debug
-check
-user <USER>
-outfile </full/path/to/file> '1>&2' to redirect to errfile
-errfile </full/path/to/file>
-pidfile </full/path/to/file>
-wait <waittime> (multiple to 10 in seconds).
-stop (stops the jsvc running associated with pidfile).

How jsvc works 1

- Downgrading user:
  - linux: setuid()/setgid() + capabilities.
  - other unixes: setgid()/initgroups().
- jsvc does something like:
  - As root:
    - init_JVM().
    - load_service. /* java_load() calls the load method */
    - downgrade user (set_caps() or set_user_group())
  - As the user USER (from -user USER parameter):
    - umask()
    - start_service. /* java_start() calls the start method */

How jsvc works 2

- 3 processes are involved:
  - Launcher process.
  - Controller process.
  - Controlled process.

- Launcher process:

```
main()
{
  fork()
  parent: wait_child(), wait until JAVA service started when the child says "I
    am ready".
  child: controller process.
}
```

How jsvc works 3

- Controller process:

```
while (fork()) {
  parent: wait_for_child.
    if exited and restart needed continue
    else exit.
  child: exit(child()). controlled process.
}
```

How jsvc works 4: Controlled process

In child(): controlled process.

init_JVM().

load_service().

start_service().

say "I am ready"

wait for signal or pool for stop

stop_service().

destroy_service().

destroy_JVM().

exit (with different codes so that parent knows if it has to restart us). while (fork()) {

parent: wait_for_child.

if exited and restart needed continue

else exit.

child: exit(child()). controlled process.

}

procrun features

- Prunmgr (GUI part):
  - //ES// Edit configuration. (serviceW.exe)
  - //MS// Monitor service (put an icon).
- Prunsrv (service/console):
  - //TS// as console application (service.exe)
  - //RS// only from ServiceManager.
  - //SS// Stop the service.
  - //US// Update service parameters.
  - //IS// Install service.
  - //DS// Delete and Stops service.

Example of a class for jsvc

- public void init(String[] arguments):
  - throws Exception
  - called by load_service()
- public void start():
  - called by start_service()
- public void stop():
  - throws IOException, InterruptedException
  - called by stop_service()
- public void destroy():
  - called by destroy_service()

Example of a class for procrun

- public void init(String[] arguments):
  - throws Exception
  - called by serviceStart()
- public void stop():
  - throws IOException, InterruptedException
  - called by serviceStop()

Mixing java and native C

- From JAVA to C:
  - public static native void toto(); (in class SimpleDaemon).
  - JNIEXPORT void JNICALL Java_SimpleDaemon_toto (JNIEnv *env, jclass class) { }
- From C to JAVA:
  - JNI_CreateJavaVM() Gives a JNIEnv *env
  - env->FindClass()
  - env->GetStaticMethodID(class)
  - env->CallStaticVoidMethod(class, method, ...).

Supported JVM's

- SUN
- IBM
- GCC GCJ (http://gcc.gnu.org/java/)
- SableVM (http://sablevm.org/)
- Jrockit.

Futur

- Use APR
- Have a single code for Unix and win32/64.
- Support more JVM like Harmony.
- Make library version.

Comments

- Any comments?

- Thank you, have Fun!