**Installing a Java Application as a Windows Service**

Ever wondered how to install a Java application as a Windows service? This article shows you how to do it.

**·**

**Jun. 27, 16 ·**[**Java Zone**](https://dzone.com/java-jdk-development-tutorials-tools-news)

**Like** [**(6)**](https://dzone.com/articles/installing-a-java-application-as-a-windows-service?edition=187564&utm_source=Daily%20Digest&utm_medium=email&utm_campaign=dd%202016-06-28)

**Comment (1)**

**Save**

[**Tweet**](https://dzone.com/articles/installing-a-java-application-as-a-windows-service?edition=187564&utm_source=Daily%20Digest&utm_medium=email&utm_campaign=dd%202016-06-28)

**5,851 Views**

*Check out this*[*8-step guide*](https://dzone.com/go?i=124041&u=http%3A%2F%2Fpages.zeroturnaround.com%2FRocket-Powered-White-Paper_Rocket-Powered-White-Paper.html%3Futm_source%3Ddzone%26utm_medium%3Djavazone_partner_resources%26utm_campaign%3Drocketpowered)*to see how you can increase your productivity by skipping slow application redeploys and by implementing application profiling, as you code! Brought to you in partnership with [ZeroTurnaround](https://dzone.com/go?i=124041&u=http%3A%2F%2Fpages.zeroturnaround.com%2FRocket-Powered-White-Paper_Rocket-Powered-White-Paper.html%3Futm_source%3Ddzone%26utm_medium%3Djavazone_partner_resources%26utm_campaign%3Drocketpowered" \t "_blank).*

It sounds like something you’d never need, but sometimes, when you distribute end-user software, you may need to install a java program as a Windows service. I had to do it because I [developed a tool](https://github.com/governmentbg/opendata-ckan-pusher) for civil servants to automatically convert and push their Excel files to the opendata portal of my country. The tool has to run periodically, so it’s a prime candidate for a service (which would make the upload possible even if the civil servant forgets about this task altogether, and besides, repetitive manual upload is a waste of time).

Even though there are numerous posts and StackOverflow answers on the topic, it still took me a lot of time because of minor caveats and one important prerequisite that few people seemed to have – having a bundled JRE, so that nobody has to download and install a JRE (would complicate the installation process unnecessarily, and the target audience is not necessarily tech-savvy).

So, with maven project with jar packaging, I first thought of packaging an .exe (with [launch4j](http://launch4j.sourceforge.net/)) and then registering it as a service. The problem with that is that the java program uses a scheduled executor, so it never exits, which makes starting it as a process impossible.

So I had to “daemonize” it, using [commons-daemon](https://commons.apache.org/proper/commons-daemon/procrun.html) procrun. Before doing that, I had to assemble every component needed into a single target folder – the fat jar (including all dependencies), the JRE, the commons-daemon binaries, and the config file.

You can see the full [maven file here](https://github.com/governmentbg/opendata-ckan-pusher/blob/master/pom.xml). The relevant bits are (where ${installer.dir} is ${project.basedir}/target/installer}):

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<executions>

<execution>

<id>assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<finalName>opendata-ckan-pusher</finalName>

<appendAssemblyId>false</appendAssemblyId>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-antrun-plugin</artifactId>

<version>1.7</version>

<executions>

<execution>

<id>default-cli</id>

<phase>package</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<target>

<copy todir="${installer.dir}/jre1.8.0\_91">

<fileset dir="${project.basedir}/jre1.8.0\_91" />

</copy>

<copy todir="${installer.dir}/commons-daemon">

<fileset dir="${project.basedir}/commons-daemon" />

</copy>

<copy file="${project.build.directory}/opendata-ckan-pusher.jar" todir="${installer.dir}" />

<copy file="${project.basedir}/install.bat" todir="${installer.dir}" />

<copy file="${project.basedir}/uninstall.bat" todir="${installer.dir}" />

<copy file="${project.basedir}/config/pusher.yml" todir="${installer.dir}" />

<copy file="${project.basedir}/LICENSE" todir="${installer.dir}" />

</target>

</configuration>

</execution>

</executions>

</plugin>

You will notice the installer.bat and uninstaller.bat which are the files that use commons-daemon to manage the service. The installer creates the service. Commons-daemon has three modes: .exe (which allows you to wrap an arbitrary executable), Java (which is like .exe, but for java applications) and JVM (which runs the java application in the same process; I don’t know how exactly though).

I could use all three options (including the launch4j created .exe), but the JVM allows you to have a designated method to control your running application. The StartClass/StartMethod/StopClass/StopMethod parameters are for that. Here’s the whole installer.bat:

commons-daemon\prunsrv //IS//OpenDataPusher --DisplayName="OpenData Pusher" --Description="OpenData Pusher"^

--Install="%cd%\commons-daemon\prunsrv.exe" --Jvm="%cd%\jre1.8.0\_91\bin\client\jvm.dll" --StartMode=jvm --StopMode=jvm^

--Startup=auto --StartClass=bg.government.opendatapusher.Pusher --StopClass=bg.government.opendatapusher.Pusher^

--StartParams=start --StopParams=stop --StartMethod=windowsService --StopMethod=windowsService^

--Classpath="%cd%\opendata-ckan-pusher.jar" --LogLevel=DEBUG^ --LogPath="%cd%\logs" --LogPrefix=procrun.log^

--StdOutput="%cd%\logs\stdout.log" --StdError="%cd%\logs\stderr.log"

commons-daemon\prunsrv //ES//OpenDataPusher

A few clarifications:

* The JVM parameter points to the JVM .dll (to be honest, I’m not sure if this will work if there is no other java installation on the machine; it should)
* The StartClass/StartMethod/StopClass/StopMethod point to a designated method for controlling the running application. In this case, starting would just call the main method, and stopping would shut down the scheduled executor, so that the application can exit
* The classpath parameter points to the fat jar
* Using %cd% is risky for determining the path to the current directory, but since the end-users will always be starting it from the directory where it resides, it’s safe in this case.

The windowsService looks like that:

public static void windowsService(String args[]) throws Exception {

String cmd = "start";

if (args.length > 0) {

cmd = args[0];

}

if ("start".equals(cmd)) {

Pusher.main(new String[]{});

} else {

executor.shutdownNow();

System.exit(0);

}

}

One important note here is the 32-bit/64-bit problem you may have. That’s why it’s safer to bundle a 32-bit JRE and use the 32-bit (default) prunsrv.exe.

I then had an “installer” folder with JRE and commons-daemon folders and two bat files and one fat jar. I could then package that as a self-extractable archive and distribute it (with a manual, of course). I looked into [IzPack](http://izpack.atlassian.net/) as well, but couldn’t find how to bundle a JRE (maybe you can).

That’s a pretty niche scenario – usually we develop for deploying to a Linux server, but providing local tools for a big organization using Java may be needed every now and then. In my case the long-running part was a scheduled executor, but it can also run a jetty service that serves a web interface. Why would it do that, instead of providing a URL – in cases where access to the local machine matters. It can even be a distributed search engine ([like that](http://www.faroo.com/)) or another p2p software that you want to write in Java.

*The Java Zone is brought to you in partnership with [ZeroTurnaround](https://dzone.com/go?i=97821&u=http%3A%2F%2Fpages.zeroturnaround.com%2FRocket-Powered-White-Paper_Rocket-Powered-White-Paper.html%3Futm_source%3Ddzone%26utm_medium%3Djavazone_partner_resources%26utm_campaign%3Drocketpowered" \t "_blank). Check out this*[*8-step guide*](https://dzone.com/go?i=97821&u=http%3A%2F%2Fpages.zeroturnaround.com%2FRocket-Powered-White-Paper_Rocket-Powered-White-Paper.html%3Futm_source%3Ddzone%26utm_medium%3Djavazone_partner_resources%26utm_campaign%3Drocketpowered)*to see how you can increase your productivity by skipping slow application redeploys and by implementing application profiling, as you code!*

**Like** [**(6)**](https://dzone.com/articles/installing-a-java-application-as-a-windows-service?edition=187564&utm_source=Daily%20Digest&utm_medium=email&utm_campaign=dd%202016-06-28)

**Comment (1)**

**Save**

[**Tweet**](https://dzone.com/articles/installing-a-java-application-as-a-windows-service?edition=187564&utm_source=Daily%20Digest&utm_medium=email&utm_campaign=dd%202016-06-28)

**5,851 Views**

Topics:

SOFTWARE,WINDOWS,TARGET,JAVA,JAVA PROGRAM,SERVICE,FAT,JAR,JRE,INSTALLATION

Published at DZone with permission of Bozhidar Bozhanov , DZone MVB .

Opinions expressed by DZone contributors are their own.