
JSmooth 0.9.7 User Manual

Build 20040316-87

Rodrigo Reyes

Revision 1.0

Revision History

10 Sep 2003

First draft

RR

Table of Contents

1. Introduction	1
1.1. What is JSmooth?	1
1.2. Rationale	2
1.3. Benefits	2
2. JSmoothGen: Windows Project Editor	2
2.1. Running the JSmoothGen project editor	2
2.2. Creating your first exe	3
3. JSmoothGen structure	4
3.1. Skeleton Selection	4
3.2. The Windows Executable	5
3.3. Defining the Java Application parameters	7
3.4. Selecting the most suitable JVM	9
3.5. Configuring the JVM	10
3.6. Skeleton Properties	13
4. Command Line	14
4.1. Running jsmoothcmd.exe	14
5. Using JSmooth as an Ant Task	14
5.1. Defining the ant task	15
5.2. Using the jsmoothgen task	15
6. Frequently Asked Question	15
6.1. Wrappers Behaviour	15
6.2. Trouble & issues with wrappers	16
7. License	16
8. Third-Party libraries used	16

1. Introduction

1.1. What is JSmooth?

JSmooth is a Java Executable Wrapper. It builds standard Windows executable binaries (.exe) that contain all the information needed to launch your java application, i.e. the classpath, the java properties, the jvm version required, and so on. If Java is not installed, it helps the users by displaying a notice and optionally launching a browser to a web site where they can download a JVM.

1.1.1. Web Site

- The JSmooth web page is located at <http://jsmooth.sourceforge.net/>

- The JSmooth Project web page is hoster at <http://sourceforge.net/projects/jsmooth>.

1.2. Rationale

The deployment of desktop Java applications has been a problem since version 1 of Java. Java developers had to either bundle a JRE with their application, or let the users manage themselves the issue of installing a JVM and configuring the application. Unfortunately, none of those solutions are efficient and scalable.

The size of a JRE bundle gets bigger and bigger as the Java language evolves. While it was reasonable to bundle the 2MB JRE 1.1 with the application, bundling the 14MB JRE 1.4 increases drastically the size of the bundle. This may not be an issue for vendors that are used to sell a packaged box with a CD, but stays an open issue for open-source developer and companies distributing their software product on the internet.

Although Sun has recently introduced the Java Web Start mechanism, it does not address all the issues: users want standard windows applications, that install in the standard "program files" directory, and which they can launch by double-clicking a standard .exe.

1.3. Benefits

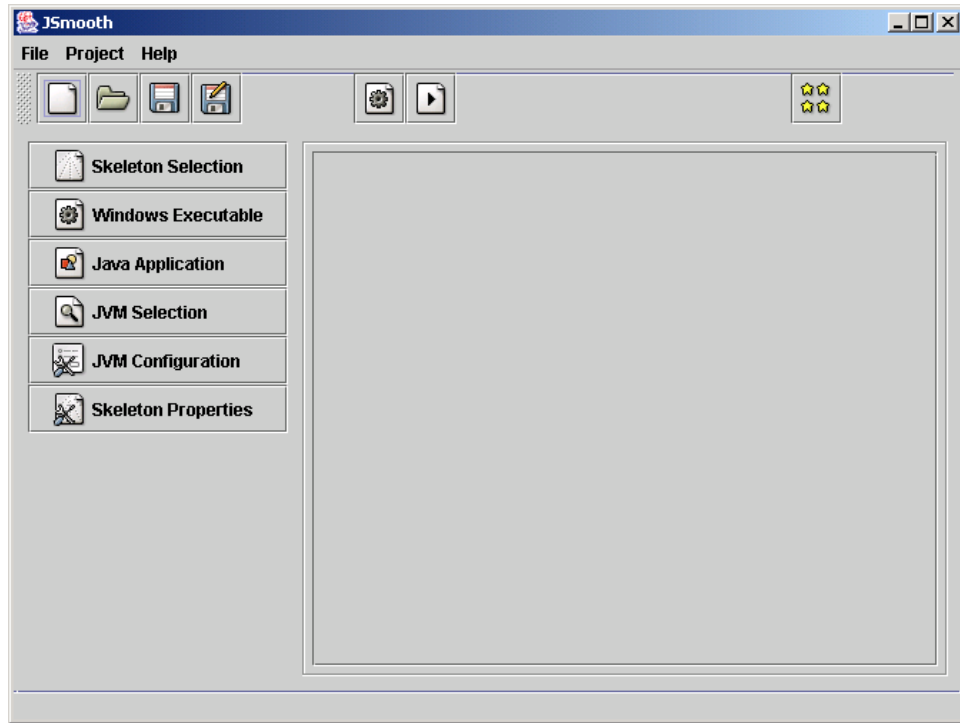
JSmooth aims at enhancing the user experience by taking care of all those issues: the java application is wrapped in an .exe, as any other Windows software. When the user clicks on the .exe, it searches the most suitable JVM installed on the computer to launch the java application. If no JVM is found, a dialog box is displayed to the users and they may be redirected to a web site.

JSmooth knows all the JVM version currently available. It knows how to pass the parameters, how to set up the memory configuration, how to manage the version constraints, and so on for every JVM version. This is the main difference between a simple .bat with "java -Xmx64m -jar myjar" which works only on the developer's computer, and a smart wrapper which adapts itself to the end-user's configuration.

2. JSmoothGen: Windows Project Editor

2.1. Running the JSmoothGen project editor

The JSmoothGen application is the project editor. It provides a graphical user interface for the configuration of all the parameters available for building a .exe.



The JSmoothGen Application

2.2. Creating your first exe

JSmooth builds a binary exe that contains all the information needed to launch your java application, such as the classpath, the java properties, the jvm version required, and so on.

2.2.1. Quick Start

The minimum data you need to start the creation of the binary executable for your application is to specify a classpath and a fully-qualified class name. You can configure both in the `Java Application` panel (add new jars or directories using the funny icon with a yellow + sign, and type the class name in the `Main Class` field).

Once done, go to the `Windows Executable` panel, and type the name of the executable in the `Executable Name` field. Do not forget to add the `.exe` suffix, for your executable name. Something like `my-app.exe` would be fine.

Of course, you could specify many more options for the java launching, but above is all what's required. Now, you are nearly done. Click on the save button (or in the `File+Save` menu) and select a filename for the project. This is an important step, because all the path stored in the file, and displayed in the GUI are relative to this file.

Once your project is saved, it is just a matter of clicking on the `Project+Create Exe`. And that's it. You can use the `Project+Run Exe` to run the program, or double-click the file name that was specified in `Windows Executable` panel, which can be found in the same directory than the project file.



Warning

You can modify the location of the every file in the project, but you must always keep this simple rule in mind:

- All the files saved and displayed in the Graphical User Interface are relative to the project file.

3. JSmoothGen structure

3.1. Skeleton Selection

The JSmooth application is based on a template system called "skeleton". Choosing a skeleton is the first step in the creation of a jsmooth executable.

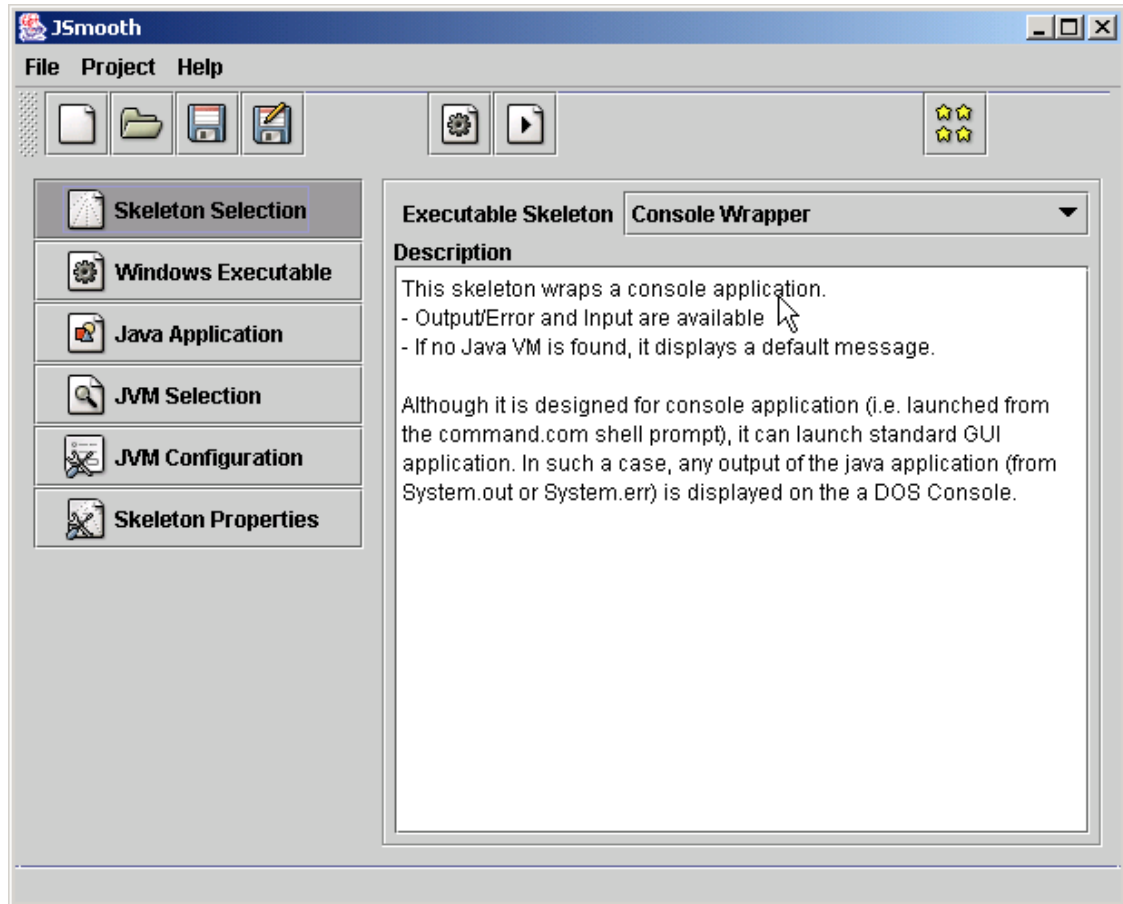
A skeleton is a template with a specific behaviour. For instance, the following skeleton are bundled with JSmooth as of version 0.9.3:

- The Console Wrapper is a skeleton designed for console applications (ie. which are run in the windows console). This skeleton takes care of passing the command line arguments to java application, it ensures the i/o are routed to the current console, and so on.

If no JVM is found, the application displays a text message to the standard output.

- The Windowed Wrapper is designed for standard Windows applications, which do not use console.

If no JVM is found, the skeleton displays a Windows OK/Cancel alert. If the user selects the OK button, the default web browser is launched on a default web page. Both the message of the alert and the URL of the web page are configurable in the Skeleton Properties panel.



The Skeleton selection panel

3.2. The Windows Executable

The windows executable created by the JSmooth wrapping system can be configured in many ways. The executable itself can be customized with the following parameters:

- The executable name is the most obvious parameter. This is simply the name of the win32 file created by JSmooth.
- The current directory can be modified to suit your needs. For instance, if you want your executable to be in a `bin\` sub-directory, but still want it to access the resources as if it were in the base directory, it may be convenient to modify the current directory to be `...`

When this parameter is specified, the wrapper simply changes the current directory with the value "as is". For instance, to change the current directory to a `res` subdirectory, write `res` in the Current Directory field.



Warning

When run as a Windows Application, the default current directory is always the directory of the executable binary. However, for a console wrapper, the default current directory is the path where the command line is currently tied.

Changing the current directory for a console application is probably not what you want. A common issue is for a console application to find resources which are found in the file system (i.e not in the jar of the application). Lots of application work this way, but Java applications, unfortunately, do not have access to the information they need: the location of the binary executable they are launched from. A work-around is to use an environment variable specific to your application. This is what ANT does, for instance.

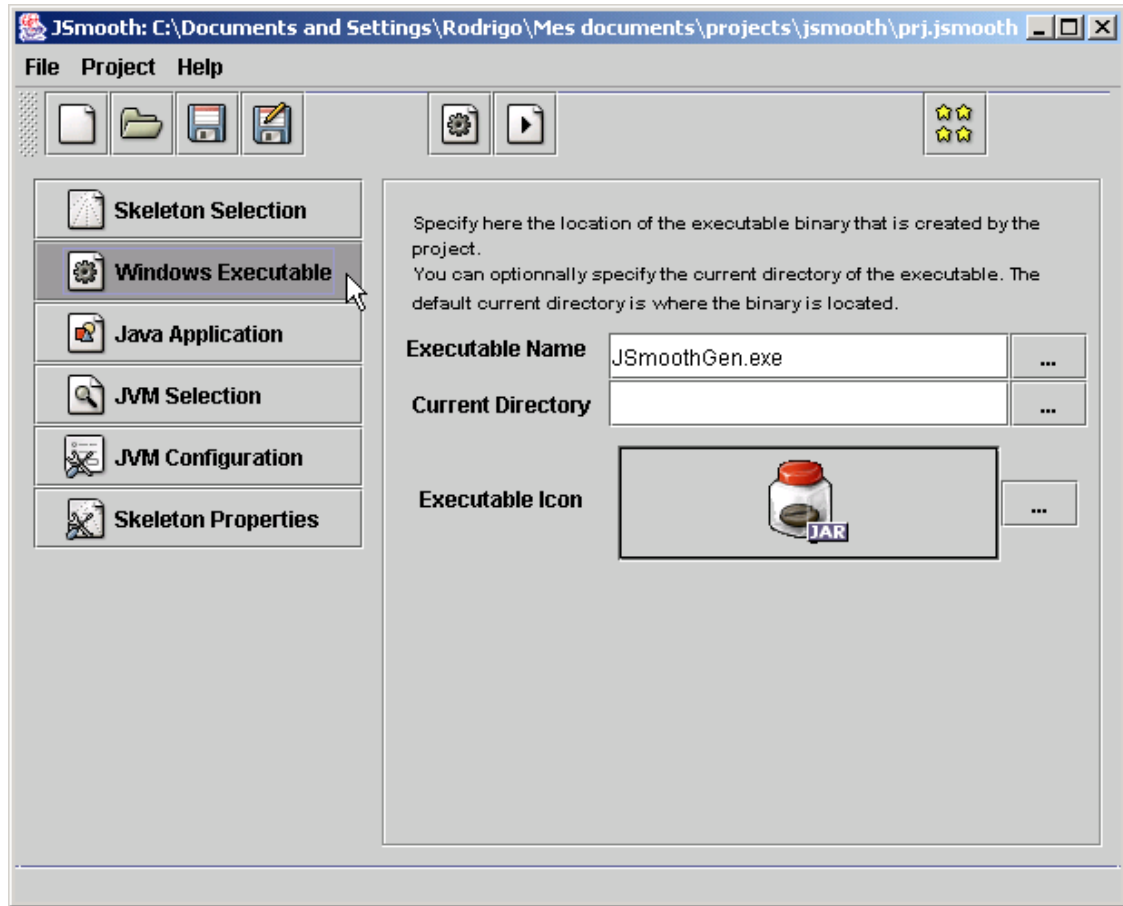
You can do something equivalent with JSmooth without using an environment variable (which may be complicated to set under windows). Instead, define a Java Property, and set `{EXECUTABLEPATH}` as value. The effect of this is to pass to your application a java property which indicates where the executable is located. Just use `System.getProperty()` call to retrieve the value.

- The executable icon field specifies the icon image that is associated to the executable, under Windows. The default configuration of JSmoothGen supports Windows Icon files (.ICO) as well as the standard types supported by the JVM (GIF, PNG, and JPG files).

The current implementation only sets up a single icon image in the executable. Here is a brief description on how JSmoothGen creates the executable icon:

- If the user selects a standard Windows icon (.ICO), it chooses 1 first 32x32 icon, then 64x64, then 16x16.
- If the user selects any other recognized image file, the image is scaled to be 32x32, and quantized to fit in 256 colors. The color reduction is rough and it is always better to use an image processing software for the job.

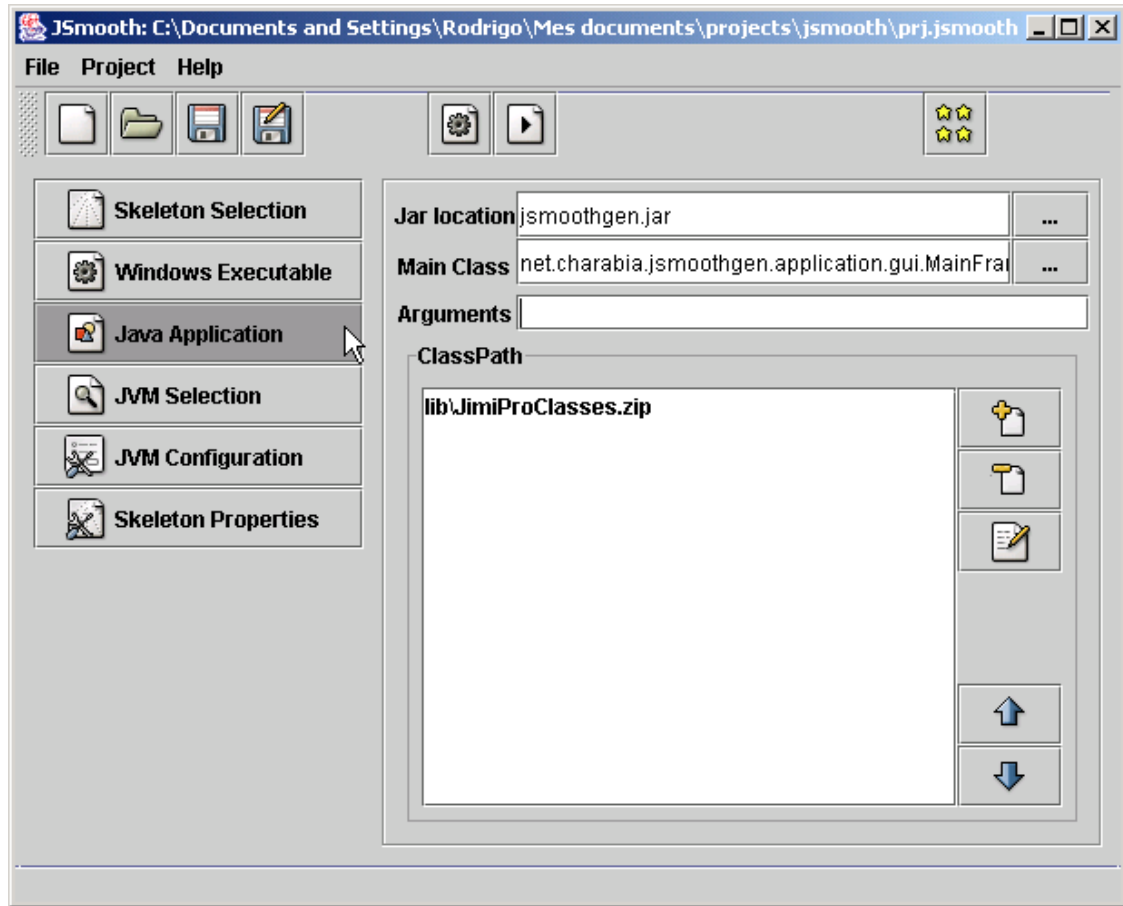
¹The Windows ICO file format can store several images in a single file. A single file often stores icons at several resolutions.



The Executable panel

3.3. Defining the Java Application parameters

This is the panel where you can specify all the parameters related to the java application itself.






The Java Application configuration panel



- The Jar Location field specifies the main JAR file of the application. There may be additional jar specified in the Classpath interface, but this one is mandatory. It contains the main class of the application, and is added as a resource to the executable created by JSmooth.
- Edit the Main Class field to specify the wrapper which class of the jar (see above) is to be considered the entry point of the program. This field is mandatory as well.
- The Arguments field allows you to specify default parameters for the application. Those arguments are passed as if they were specified on the command line, and are made available in the String array of the main method by Java.



Warning

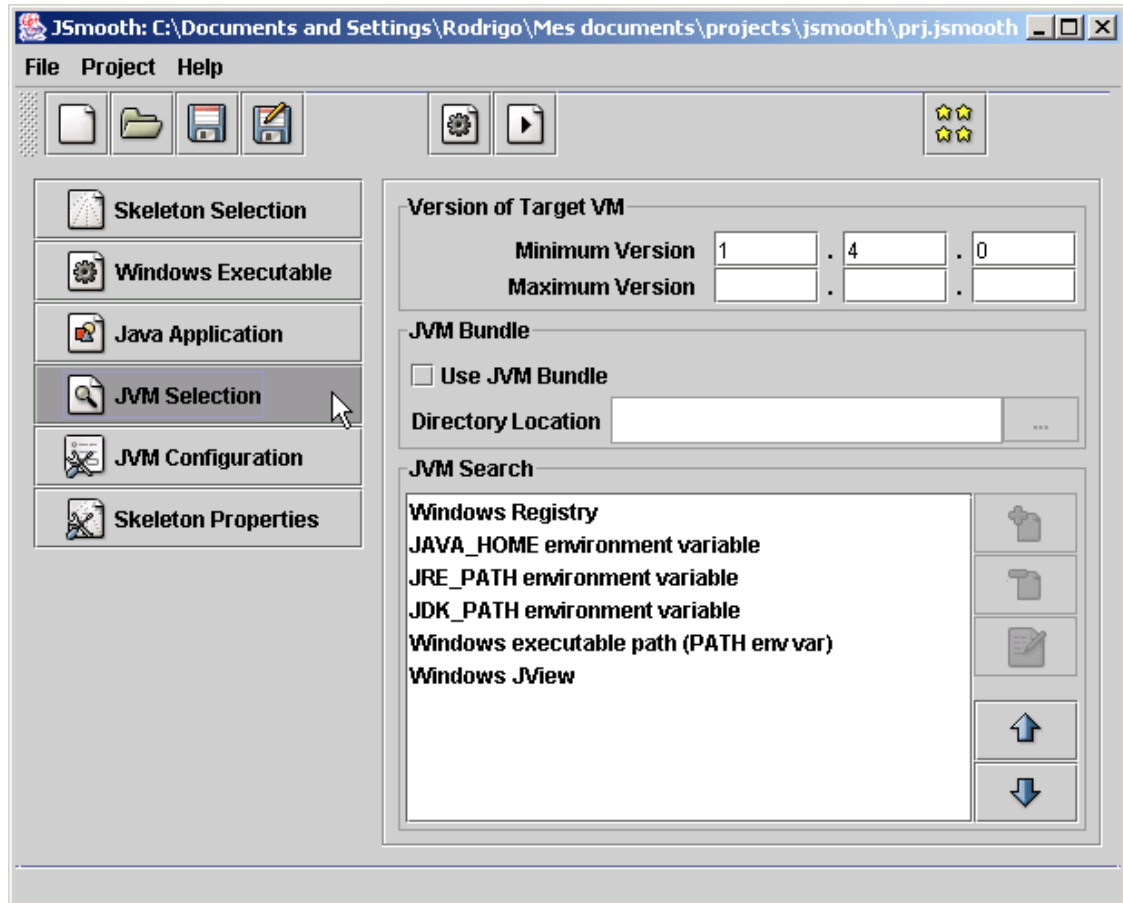
The Arguments may be overridden by some wrappers. For instance, the Console Wrapper uses the arguments from the command line instead.

- The ClassPath list proposes you to edit an ordered list of classpaths. To add a classpath, click on the  and select either a .JAR, a .ZIP or a directory using the file chooser. You can remove or edit the classpath entries with the  and  buttons.

The items can be ordered to reflect the class look-up priorities of your application. Select an item, and press either  or . The rules are simple: the higher in the list, the higher the priority in the classpath search sequence.

3.4. Selecting the most suitable JVM

Click on the JVM Selection tab to specify the Java look-up parameters.



The JVM Selection panel

3.4.1. Java Version

If your application requires a minimum or a maximum version of Java to run, the Version of Target VM subpanel will probably be of high interest for you.

Just leave the field empty if you do not wish to any constraint on the JVM version required for your application.

To specify a version, either as a minimum or a maximum, you must specify the major, minor, and sub-minor version number you want to set the constraint on. You don't need to specify all of those three numbers, but remember that if you leave it empty, it is considered as 0.

The values are all inclusive. For instance, if you specify 1.4 as a maximum means that the java wrapper accepts 1.4.0 JVM, but not 1.4.1.

If your application runs exclusively with JVM 1.2 and 1.3, but not 1.1 nor 1.4 or above, just specify 1.2 for the minimum version, and 1.3.99 as a maximum version.

3.4.2. JVM Bundle

Even if the wrappers of JSmooth provide a reasonable user experience for the users who do not have Java installed, you may wish to bundle a JRE with your application. In such a case, tick the Use JVM Bundle checkbox, and specify the directory location of the JRE.



Warning

What the "JVM Bundle" option really specifies, is a path (relative to the generated EXE) where a JRE can be found. This is NOT an option to bundle a JRE in the EXE, as many people may think.

For the option to work correctly, you have to put a JRE in a directory near the EXE (generally in a sub-directory called "jre" or whatever). Once the exe is generated, it will FIRST try to locate the JRE at the location mentioned. If it can't be found there, then it will fallback in the normal jre look-up mode (search for a jre or a jdk in the Windows registry or in commonly-used environment variables).

Therefore, if you use the "Bundle JVM" option, you'll need to install yourself the JRE at the same relative path to the EXE.



For instance, take the following example:

```
<codelisting> +myprog/ |- myjar.jar |- lib/ + mylib1.jar + mylib2.jar |- jre/ + [full jre stuff here] |- my-  
exe.exe </codelisting>
```

In this case, the generated myexe.exe will try to use first the jre in the "jre" sub-dir. To deploy it, either simply zip all the "myprog" directory, or tell your favorite installer to set-up the jre directory in the same configuration as in your original folder tree.

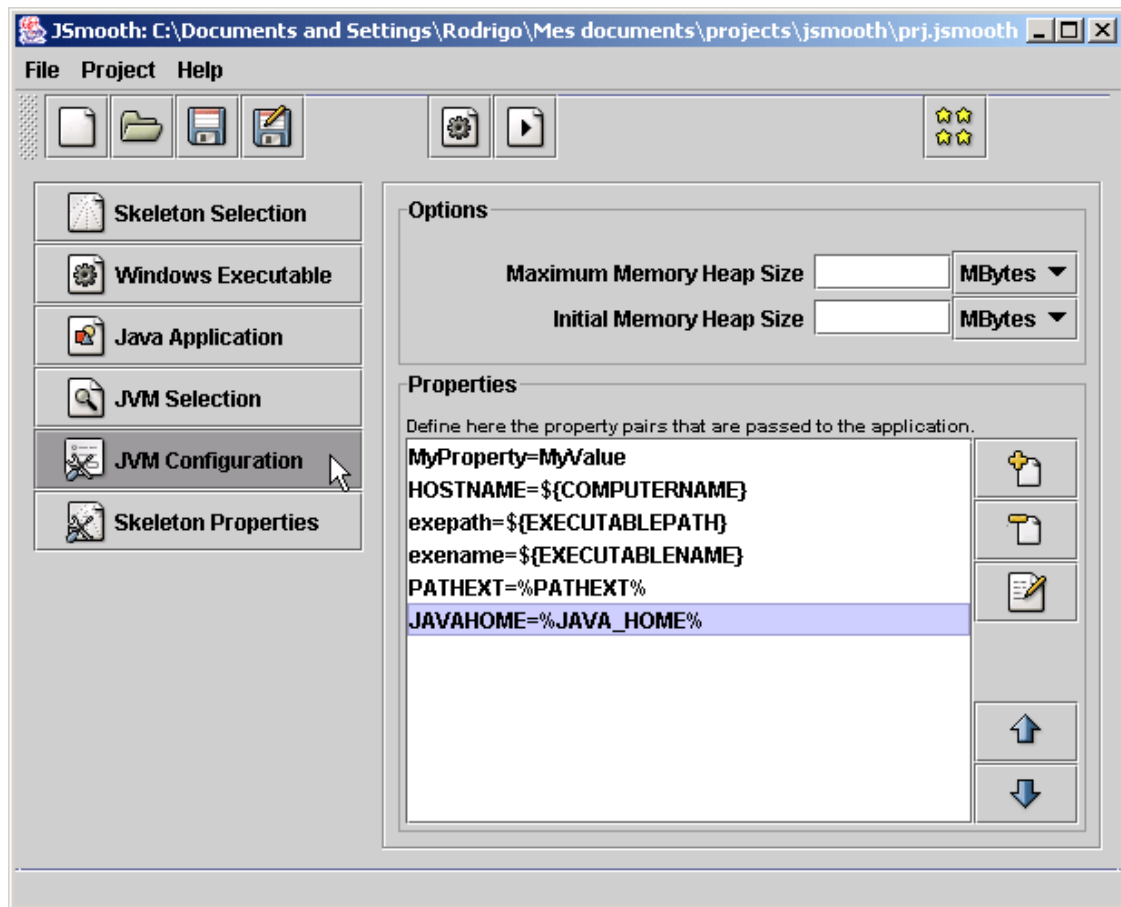
3.4.3. The JVM Search sequence

The wrappers are able to detect most, if not all, the JVM available on the end-user computer. Before launching the java application, they try to find a JVM using a preference order. The default is to use the Windows registry to look-up the JRE that have been installed, then use some environment variable, and finally try to use Microsoft's JView.

This search sequence is fine for most configuration, however you can still modify it to best suit your needs. Use the  and  button to modify the priorities..

3.5. Configuring the JVM

The JVM Configuration offers the possibility to specify the parameters passed to the Java Virtual Machine when launching your application.




The JVM configuration panel

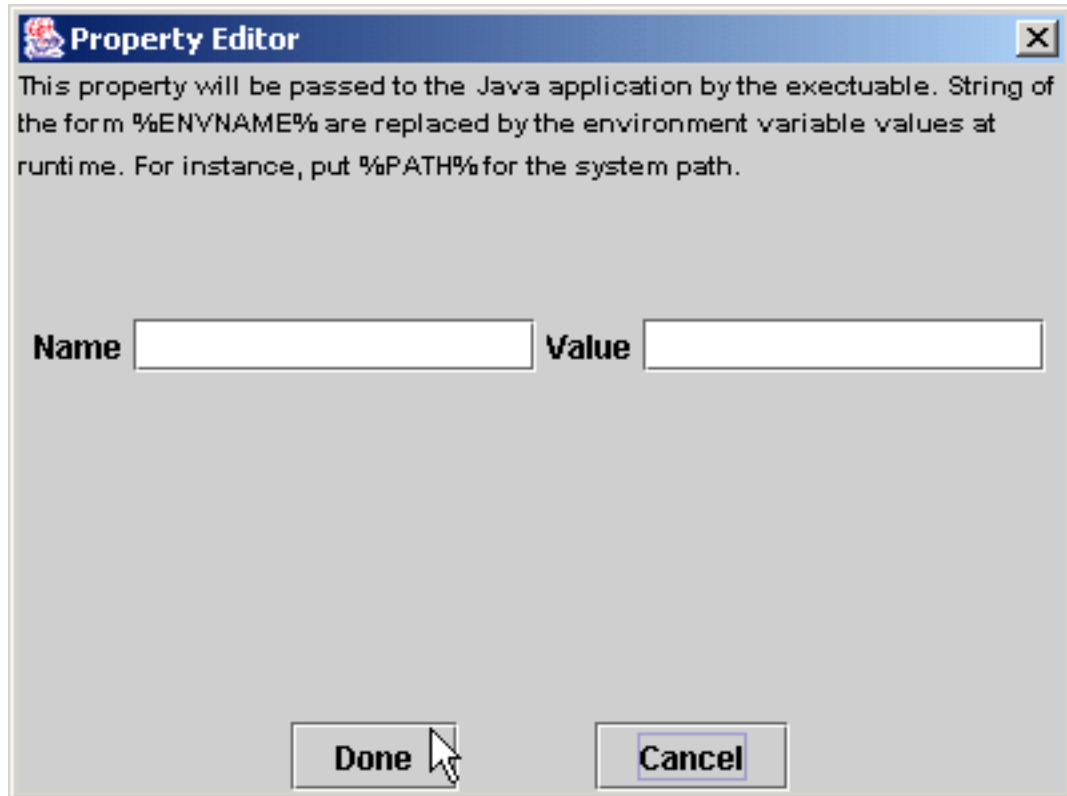
3.5.1. Options

The Options panel provides editable fields to configure the memory configuration of the JVM. Specify the numeric value in the text field, and the unit in the combo box.

3.5.2. Java Properties

Java properties are name/value pairs that are passed to the JVM, and are accessible by java applications using the `System.getProperty()` method.

To add a new Java Property, click on the . A dialog box pops up.



Specify the name and the value for this property, and click on the Done button.

3.5.2.1. Special Values

You can use the Java Property mechanism to pass special values to your java app.

- **Environment Variable:** If you pass a string of the form %ENV% in the value field of the property, the wrappers will replace the string with the content of the ENV environment variable.

For instance, to pass the content of the JAVA_HOME environment variable, add %JAVA_HOME% in the Value field.

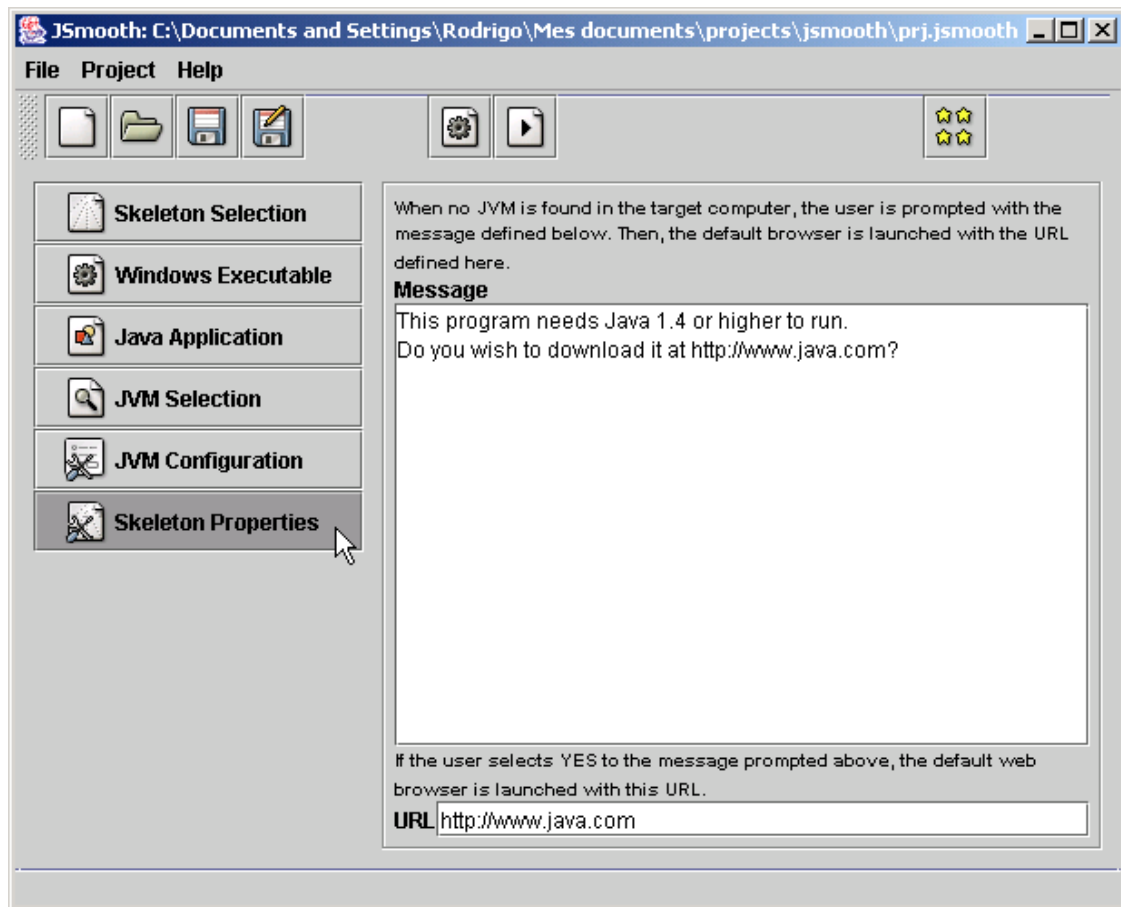
- JSmooth also makes some special variable accessible for your application.

Form	Meaning
<code>\${EXECUTABLEPATH}</code>	Replaced by the path to the executable binary. For instance, if the executable binary launched is located at c:/program files/jsmooth/test.exe, this variable is replaced with c:/program files/jsmooth
<code>\${EXECUTABLENAME}</code>	Replaced by the name of the executable binary. For instance, if the executable binary launched is located at c:/program files/jsmooth/test.exe, this variable is replaced with test.exe
<code>\${COMPUTERNAME}</code>	Replaced by the name of the computer.

<code>\${VMSELECTION}</code>	<p>Replaced by the name of the selection method used to spawn the JVM. The value replaced is typically one of the following values:</p> <ul style="list-style-type: none">• <code>registry</code>, when the VM location is found in the registry• <code>jdkpath</code>, when the VM is found in the <code>%JDK_HOME%</code> variable• <code>jrepath</code>, when the VM is found in the <code>%JRE_HOME%</code> variable• <code>javahome</code>, when the VM is found in the <code>%JAVA_HOME%</code> variable• <code>jview</code>, if the VM is JView• <code>exepath</code>, if the VM was found in the windows path• <code>bundled</code>, if the VM bundled with the application is used
<code>\${VMSPAWNTYPE}</code>	<p>This variable is replaced by one of the following values:</p> <ul style="list-style-type: none">• <code>JVMDLL</code> if the JVM is spawned using a DLL loading and instantiation• <code>PROC</code> if the JVM is spawned in a new process using the Windows' <code>CreateProcess</code> function

3.6. Skeleton Properties

This panel is specific to the wrapper selected in the Skeleton Selection panel.



The Skeleton-specifics properties panel

4. Command Line

4.1. Running jsmoothcmd.exe

You can build a wrapped java application using the `jsmoothcmd.exe` command. However, note that it only takes as argument a project file created by the JSmoothGen graphical application. In other words, you can't create a project from scratch using only the `jsmoothcmd` program, which is provided only as a convenience for scripting purposes.

Usage

```
jsmoothcmd.exe [myproject.jsmooth]
```

The `.jsmooth` suffix may be omitted. The application either creates the executable binary successfully, or returns an error message.

5. Using JSmooth as an Ant Task

JSmooth can also be used as an ant task. This permits using it as a tool integrated in a standard ant deployment chain.

5.1. Defining the ant task

This is the first step to be able to use jsmooth in your ant scripts. You need to add a `taskdef` tag that provides to ant the information required to create the `jsmoothgen` task.

```
<taskdef name="jsmoothgen"
  classname="net.charabia.jsmoothgen.ant.JSmoothGen"
  classpath="path/to/jsmoothgen-ant.jar"/>
```

If your jsmooth project uses the Sun's JIMI library to recognize .ico files, you must add it to the classpath as well.

5.2. Using the jsmoothgen task

The task takes two mandatory parameters:

- `project`: a reference to the project file.
- `skeletonroot`: a reference to a directory where the skeletons can be found. Be careful, this is not the directory where the skeleton used by the project can be found, be rather the directory one level up.

Here is a sample, extracted from the jsmooth build.xml file:

```
<jsmoothgen project="jsprj/jsmoothgen.jsmooth"
  skeletonroot="${dist}/skeletons"/>
```

6. Frequently Asked Question

6.1. Wrappers Behaviour

- 6.1.1. How does a wrapper extract the jar file?

The jar file extraction is wrapper-specific. However, all the wrapper provided with JSmooth as of version 0.9.3 extract it in the default temporary directory. This is the standard behaviour expected by Windows application.

- 6.1.2. What happens to the extracted jar file when the java application exits?

Whenever possible, the wrappers delete the file on exit. Note however that this is not always possible. For example, an instance of a JVM created using the JVM DLL does not unload cleanly. This behaviour is documented by Sun, and there is no known work-around for it.

Both Windowed and Console wrapper always prefer the JVM instantiation methods that allow them to delete the temp jar after the application exits.

Note however that deleting the files in the windows TEMP directory is not required for Windows applications, and most application just leave them, letting the operating system manage it. However, MS Windows deletes those temp files only when the disks are running low in available space.

6.1.3.

Jsmooth allows only one embedded jar in the executable binary. Does it mean that all the java application must be exhaustively set up in this jar ?

No. While the jsmooth wrappers require a jar to bootstrap the java application, you can reference other jar files in a jsmooth project. Those jar files will be accessible to your executable binary, provided the path specified is found at runtime.

It is often convenient to put the main jar in the executable, but you can just put a small jar which purpose is just to bootstrap the application (for instance by displaying a splash screen with specific information and calling another class).

6.2. Trouble & issues with wrappers

6.2.1.

The wrapped applications complain about a "MSVCRT.DLL not found" on Windows 95?

The MSVCRT.DLL file is required by the wrapped application but may be missing on some Windows 95 computers that have never been upgraded. This file is often installed by third-party application, but it may happen that your users do not have it installed.

To solve this issue, either ask your user to upgrade to a decent OS, or just bundle this MSVCRT.DLL along with your application.

7. License

All the JSmooth project is distributed under the terms of the GNU General Public License. Please read the License.txt file that comes with the package. Alternatively, you can find additional information on the GNU GPL license on the GNU Web Site [<http://www.gnu.org>]

This license applies to all the files of the project, but not on the generated executable. This means that you are free to generate executable wrappers for proprietary software and distribute them without applying the terms of the GPL to them.

Of course, this is the one and only exception. All other kind of distribution of any file of the JSmooth package must conform with the terms of the GNU General Public License.

8. Third-Party libraries used

This product includes software developed by L2FProd.com (<http://www.L2FProd.com/>). The l2fprod-common library is licensed under the terms of Apache Software License. Source code can be found at the following location: <https://l2fprod-common.dev.java.net/>

JSmooth includes the JOX library. JOX is a set of Java libraries that make it easy to transfer data between XML documents and Java beans. It is distributed under the terms of the Lesser GPL. Source

code for JOX is available at the library's web site: <http://www.wutka.com/jox.html>.

JSmooth includes the DTDParser library. It is distributed under the terms of either an Apache-style license or the Lesser GPL license. Source code for is available at the library's web site: <http://www.wutka.com/dtdparser.html>.