Getting Started

Getting started

Overview

To begin with, you should know how IzPack is organized if you want to use it. Let's go into the directory where you have installed IzPack on your machine. There are 3 text files and a set of directories. The most important for the moment are bin/ doc/ sample/. If you are reading this, you already know that doc contains this documentation 🙂

So let's go into bin/. The icons/ directory contains some directories for your system, in case you would like an icon to launch a component of IzPack . But the most important things you can see in bin are the compile scripts (in both unix* and windows formats). Compile is used to compile a ready-to-go xml installation file from a command-line context or from an external tool.

Note : these scripts can be launched from anywhere on your system as the installer has customized these scripts so that they can inform IzPack of where it is located.

Installation of IzPack

First go get the latest stable version of IzPack from: http://izpack.org/downloads

If needed download the Latest Java Run Time from Sun's website http://java.sun.com/. You should get the JRE if you intend to ONLY run the installer and get the SDK if you're willing to compile as well.

On Windows

Don't forget to set up the environment variables:

If using the SDK:

```
set JAVA_HOME="C:\j2sdk1.4.2_04"
set JRE_HOME=%JAVA_HOME%/jre
set CLASSPATH=%JAVA_HOME%/bin;%CLASSPATH%
set
PATH=%JAVA_HOME%/bin;%JRE_HOME%/bin;%PATH%
```

This is obviously assuming that SDK has been installed to "C:\j2sdk1.4.2_04"

If using the JRE:

```
set JAVA_HOME="C:\Program
Files\Java\j2re1.4.2_05"
set CLASSPATH=%JAVA_HOME%/bin;%CLASSPATH%
set PATH=%JAVA_HOME%/bin;%PATH%
```

This is obviously assuming that JRE has been installed to "C:\Program Files\Java\j2re1.4.2_05" Once this is done, you can install IzPack using the following command:

```
java -jar izpack.jar
```

Where izpack.jar is the latest release you downloaded from IzPack website.

On UNIX/Linux

If needed download the Latest Java Run Time from Sun's website http://java.sun.com/. You should get the JRE if you intend to ONLY run the installer, but you should get the SDK if you're willing to compile as well.

If using the SDK:

```
export JAVA_HOME=/usr/java/j2sdk1.4.2_06
export JAVA_JAR=/usr/java/java_jar
export JRE_HOME=/usr/java/j2sdk1.4.2_06/jre
export CLASSPATH=/usr/java/j2sdk1.4.2_06/bin
export
PATH=/usr/java/j2sdk1.4.2_06/bin:/usr/java/j
2sdk1.4.2_06/jre/bin:$PATH
```

This is obviously assuming that java has been installed to /usr/java/j2sdk1.4.2_06

If using the JRE:

```
export JAVA_HOME=/usr/java/j2re1.4.2_05
export CLASSPATH=$JAVA_HOME/bin:$CLASSPATH
export PATH=$JAVA_HOME/bin:$PATH
```

This is obviously assuming that JRE has been installed to "/usr/java/j2re1.4.2_05" You can put them into any script launched at startup if you don't want to have to do it everytime. For example, .bashrc of your user, so that whenever you'll start a bash console the variables will be set.

To verify that the environment is correct, type SET in the command prompt and check if those variables are set before running any compilation.

Then you install IzPack using the following command:

```
java -jar izpack.jar
```

By default it will be installed in /usr/local/IzPack. Therefore you can create two scripts, one for compiling your code and the second to execute the installer.

Compile.sh:

```
#!/bin/sh
/usr/local/IzPack/bin/compile
/yourpath/Install.xml -b /yourpath -o
/yourpath/yourjaroutput.jar -k standard
```

Install.sh:

```
#!/bin/sh
java -jar yourjaroutput.jar
```

BUGS and TROUBLESHOOTING

1. This is assuming that you're current Unix/Linux allows the use of the server X. In cas it doesn't here is a way to install IzPack using cygwin (thanks to Shrish Buradkar and Bartz Klaus for this trick):
   Install cygwin on a remote machine. Cygwin can be downloaded from http://www.cygwin.com/ Firstly, start the XWindows server on your PC. This could be done by using the startxwin-multiwindow batch file or running

/usr/X11R6/bin/startxwin.sh From the cygwin Xterm, type xhost + Then telnet to the remote UNIX/Linux machine and set the DISPLAY to your PC. So after you have logged into the remote machine, do export DISPLAY=pc-ip-adress:0.0 xterm & java -jar installer.jar This should do the job by displaying an xterm from the remote machine onto yor PC.

2. Normally launching packages created by IzPack under Gnome, KDE or XFCE works fine. If when trying to launch a pack you receive this error message:

```
Exception in thread "main" java.lang.InternalError: Can't connect to X11
window server using ':0.0' as the value of the DISPLAY variable.
        at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
        at
sun.awt.X11GraphicsEnvironment.<clinit>(X11GraphicsEnvironment.java:134)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:141)
        at
java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment(GraphicsEnvironm
ent.java:62)
        at java.awt.Font.initializeFont(Font.java:308)
        at java.awt.Font.<init>(Font.java:344)
        at
com.izforge.izpack.gui.IzPackMetalTheme.createFont(IzPackMetalTheme.java:
62)
        at
com.izforge.izpack.gui.IzPackMetalTheme.<init>(IzPackMetalTheme.java:52)
        at
com.izforge.izpack.gui.IzPackKMetalTheme.<init>(IzPackKMetalTheme.java:59)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native
Method)
        at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAc
cessorImpl.java:39)
        at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConst
ructorAccessorImpl.java:27)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:274)
        at java.lang.Class.newInstance0(Class.java:308)
        at java.lang.Class.newInstance(Class.java:261)
        at
com.izforge.izpack.installer.GUIInstaller.loadLookAndFeel(GUIInstaller.ja
va:297)
        at
com.izforge.izpack.installer.GUIInstaller.<init>(GUIInstaller.java:100)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native
Method)
        at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAc
cessorImpl.java:39)
        at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConst
ructorAccessorImpl.java:27)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:274)
        at java.lang.Class.newInstance0(Class.java:308)
        at java.lang.Class.newInstance(Class.java:261)
        at com.izforge.izpack.installer.Installer.main(Installer.java:47)
```

Then, it's most probably the fact that in some distribution the console and environment variables are "erased" when switching between users. So, you can type 'su -' in order to obtain all commands. With 'su -' $DISPLAY variables is erased and all X11 connections is refused. So, a best and fast practice in this way is:

1. Log in your system by user
2. In shell type: '$ echo $DISPLAY'
3. the result seems to be ':0.0'. If the response isn't there you can type:
4. '$ export $DISPLAY=":0.0"'
5. Now type $su for a "normal" alias by root.
6. Run: $ java -jar "package.jar"

First Compilation

Now you probably can't wait to build your first installer. So go on open a command-line shell and navigate to sample/. The following should work on both unix* and windows systems. for the latter, just change the path separator (slash '/') to a backslash. So type ($ is your shell prompt !) :

```
$ ../bin/compile install.xml -b . -o
install.jar -k standard
 (installer generation text output here)
$ java -jar install.jar
```

There you are! The first command has produced the installer and the second one did launch it.

How to develop and debug IzPack using Eclipse

(thanks to Bartz Klaus)

Here are the steps needed to develop and debug IzPack with Eclipse:

1. IzPack Installation
   Install the latest stable release of IzPack with the sources ! For more details see the section "IzPack Installation".
2. Custom class sources and build.xml
   Put your custom class sources under %IZPACK_HOME%srclib may be %IZPACK_HOME%srclibcomizforgeizpackpanelsMyPanel.java Add a create rule into %IZPACK_HOME%srcbuild.xml under target "build.panels"
3. Eclipse
   You can get Eclipse from http://www.eclipse.org/downloads/index.php
4. Create IzPack project
   Select File > New > Project... Java > Java Project > next > give a project name like "IzPack" deselect "Use default" (2.x) or select "Create project at external location" (3.x) Browse to %IZPACK_HOME%srclib select it Next > In "Libraries" select "Add External JARs..." select ant.jar and jakarta-regexp-1.3.jar from %IZPACK_HOME%lib Finish
5. Debug compile (create installation)
   Select Run > Debug... Java Application New give a name e.g. "CompileMyInstall" select in "Main" the project "IzPack" select as "main class" "Compile" (from package com.izforge.izpack.compiler) As "Program arguments" put in (for %SOME_THING% use your local value)
   %SRC_ROOT%%CONFIG_SUBPATH%install.xml -b %SRC_ROOT% -o %INSTALLER_DEST%install.jar
   As "VM arguments" put in "-DIZPACK_HOME=n:homebartzkauworkxt150_forIzPackizpack-src"
   No you can debug the compiling of your installation.
6. Debug installation

Compile your installation; now you have %INSTALLER_DEST%install.jar Run > Debug... Java Application New give a name e.g. "InstallMyInstall" select in "Main" the project "IzPack" select as "main class" "Installer" (from package com.izforge.izpack.installer) as "VM arguments" use -DTRACE=true select the tab "Classpath" select "User classes" (2.x) or "User Entries" (3.x) select "Add External JARs..." select %INSTALLER_DEST%install.jar (may be, that's the trick...) install.jar must be under the project entry

BUGS and TROUBLESHOOTING

If you get this error when running the application could not create shortcut instance

```
java.lang.Exception: error loading library
at com.izforge.izpack.util.Librarian.loadLibrary(Librarian.java:249)
at com.izforge.izpack.util.os.ShellLink.initialize(ShellLink.java:461)
at com.izforge.izpack.util.os.ShellLink.<init>(ShellLink.java:349)
at
com.izforge.izpack.util.os.Win_Shortcut.initialize(Win_Shortcut.java:79)
at
com.izforge.izpack.panels.shortcut.ShortcutPanel.<init>(ShortcutPanel.jav
a:473)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown
Source)
at java.lang.reflect.Constructor.newInstance(Unknown Source)
at
com.izforge.izpack.installer.InstallerFrame.loadPanels(InstallerFrame.jav
a:203)
at
com.izforge.izpack.installer.InstallerFrame.<init>(InstallerFrame.java:16
0)
at
com.izforge.izpack.installer.GUIInstaller.loadGUI(GUIInstaller.java:391)
at com.izforge.izpack.installer.GUIInstaller.<init>(GUIInstaller.java:128)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown
Source)
at java.lang.reflect.Constructor.newInstance(Unknown Source)
at java.lang.Class.newInstance0(Unknown Source)
at java.lang.Class.newInstance(Unknown Source)
at com.izforge.izpack.installer.Installer.main(Installer.java:62)
```

then it means you forgot to put the shelllink.dll into the correct source folder in that case, copy %IZPACK_HOME%binnativeizpackShellLink.dll to %IZPACK_HOME%srclibcomizforgeizpackutilosShellLink.dll

Now you can debug the installation. With 2.x you can edit on demand, if %INSTALLER_DEST%install.jar is shown in "Classpath" under the project. With 3.x it seems so, that always the JAR will be loaded; therefore the contents of install.jar and the sources should be synchron.

7. Debug uninstallation

   Install your installation to %INSTALL_PATH% Run > Debug... Java Application New give a name e.g. "UninstallMyInstall" select in "Main" the project "IzPack" select as "main class" "Uninstaller" (from package com.izforge.izpack.installer) as "VM arguments" use -DTRACE=true select the tab "Classpath" select "User classes" (2.x) or "User Entries" (3.x) select "Add External JARs..." select %INSTALL_PATH%Uninstalleruninstall.jar uninstall.jar must be under the project entry

Now, you can debug your uninstallation. Don't worry if you get first a NullPointerException in SelfModifier. This should be; it is a hint, that the class files of your eclipse session are used. If debugging not working, look whether: there is a fresh installation the uninstall.jar is in the "Classpath" tab under the project entry

The IzPack architecture

Now that you have packaged your first installer, it's time for you to understand how the whole thing works.

The compilation system

The compilation system is quite modular. Indeed, you can use the compiler in 2 ways :

- from a command-line
- from jakarta ant

The compiler takes as its input an xml installation file that describes (at a relatively high-level) the installation. this file contains detailed information such as the application name, the authors, the files to install, the panels to use, which resources to load and much more.

The compiler can generate different kinds of installers, but this information is not located inside the xml file as it is not were it should be. On the contrary, this is a compiler parameter.

The compilation options for a command-line installer are the following:

-? Gives a list of the available options. -b Specifies the base path, ie the one that will be used to resolve the relative paths. if your xml file contains absolute paths, specify it to an empty string (-b ""). -k Specifies the installer kind, for instance most users will want standard here. -o Specifies the resulting installer jar file name.

How an installer works

An installer presents its panels to the end-user. for instance, there is one to select the packages, one to prompt for the license agreement, one to select the installation path and so on. You have a choice from a variety of panels to place in the installer. For example, you can choose between a plain text and a html text panel for the license agreement. also, if you don't want of the hellopanel, you just don't include it.

It is very important to understand that some of the panels may need extra data. for instance, the license agreement panel needs the license text. A simple approach to specify such data would have been to add as many xml tags as needed for each panel. However, this makes the xml file too specific and not easy to maintain. The approach that has been chosen is to put the data in files and we call these files resource files. They are specified with a unique xml tag. this is a much cleaner approach.

You might wonder how your files are packaged. They can be grouped in packs. For instance, you can have one pack for the core files, one for the documentation, one for the source code and so on. In this way, your end-users will have the choice to install a pack or not (provided that the pack they don't want to install is not mandatory). Inside the jar file (which is a zip file), a sub directory contains the pack files. Each pack file contains the files that are part of it. Could we do it simpler ? 🙂

The different kinds of installers

There are 2 kinds of installers available :

- Standard : a single-file ready-to-run installer
- Web : a web based installer (pack data is located on an http server, and the installer retrieves it at install time (see section 3.6))

Installers for older vm versions

By default the installer will be made for the current most used version of the java runtime environment. It is possible to create an installation that is runable with an older vm version.

What version is used can be detected in the ant properties file that is used to build izpack. It is izpackroot/src/ant.properties. The value of the property "source" determines the vm version.

If compatibility to older versions is needed, a recompilation of the jar files of the izpack system should be done. For this the sources of izpack and an ant installation are needed. the sources of izpack are selectable at installation time of izpack. Before a recompilation of all can be triggered, the version of byte code should be changed. This can be done simple by changing the "source" entry in izpackroot/src/ant.properties to the needed value. The recompilation should be performed with the current most used vm version because there are classes of it referenced in the izpack code. Usage of an older vm version at installation time will be possible because the classes of the newer vm version are only used after a vm version check. Of course, some features of izpack will be missing at using an old vm version. To recompile izpack go into izpackroot/src. Use a current jdk (not jre) for this. call

```
ant clean
```

followed by

```
ant all
```

then all jar files in izpackroot/lib, izpackroot/bin/panels and izpackroot/bin/customactions should be recompiled with the selected source version.