

# The IzPack tutorial

Getting started with a basic installer for your software.

Julien Ponge<sup>1</sup>

<sup>1</sup><julien@izforge.com>

<http://www.izforge.com/>

IzPack project founder and current maintainer.

11th January 2005





Copyright © 2004, 2005 Julien PONGE - All Rights Reserved.

This work is licensed under the *Creative Commons Attribution - NonCommercial - ShareAlike License*. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

# Outline

- 1 Introduction
  - The IzPack project
  - Technical overview
- 2 Making an installer
  - Preliminary steps
  - Creating the installation files
  - Building the installer
- 3 Conclusion



# Outline

- 1 Introduction
  - The IzPack project
  - Technical overview
- 2 Making an installer
  - Preliminary steps
  - Creating the installation files
  - Building the installer
- 3 Conclusion

# IzWhat ?

## Fast facts:

- An open-sourced Java™ -based cross-platform installer generator.
- Published under the GNU GPL license.
- Project started in 2001 by myself, now developed with the help of several developers and contributors (`Thanks.txt` contains about 90 lines).
- Used by various companies and projects around the world (see the references page).
- Available for about 22 languages.
- One of the most active projects at BerliOS.

# Features

- Cross-platform (tested on Win32, Mac OS X, Linux/i386 and FreeBSD/i386).
- XML-based, modular and extensible (you choose what your installer will be made of).
- Integrates with Jakarta Ant.
- Can create shortcuts for Win32 and X11 (FreeDesktop.org-compliant environments and window managers).
- Not dependent on native code (but can use it in a smart way).
- Creates uninstallers.
- Can get user input, substitute tokens in files, call scripts and much more...

# Project life

- Online resources:
  - hosted on my website at <http://www.izforge.com/izpack/>
  - BerliOS hosts the developer tools (CVS, SVN, bugs tracking, wiki, mailing-lists, file releases, FTP, ...).
- The development is very open and contributions are always welcome.
- Major releases (ex: 3.7.x) are maintained in branches while the development occurs in CVS HEAD.
- Minor releases occur depending on fixes inclusion.
- Major releases happen depending on new features inclusion.

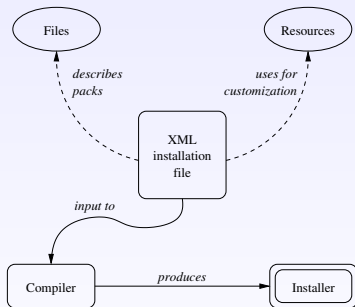
# Outline

- 1 Introduction
  - The IzPack project
  - **Technical overview**
- 2 Making an installer
  - Preliminary steps
  - Creating the installation files
  - Building the installer
- 3 Conclusion



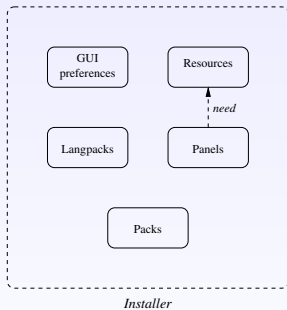
# IzPack

- An installer is described by an *XML installation file* that:
  - arranges files into *packs*
  - can be customized by some resources (depending on what you choose).
- The compiler takes the XML file as its input to build an installer as an *executable Jar* archive.



# Installers

- An installer contains:
  - the real files in packs
  - the langpacks.
- An installer offers a set of panels that define the steps to perform an installation.
- Resources can be needed by panels and GUI preferences can change the look depending on the OS (size, L&F, ...).

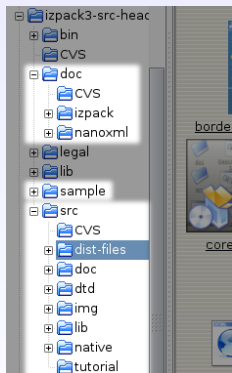


# Outline

- 1 Introduction
  - The IzPack project
  - Technical overview
- 2 Making an installer
  - Preliminary steps
  - Creating the installation files
  - Building the installer
- 3 Conclusion

# Laying out the files and folders

- Put your files in a folder. Try to make it easy to split the files tree into packs (for instance put the files of a pack into a small set of subfolders).
- Think about which files and folders your packs will be made of.
- Decide which packs will be mandatory and which packs will be optional.

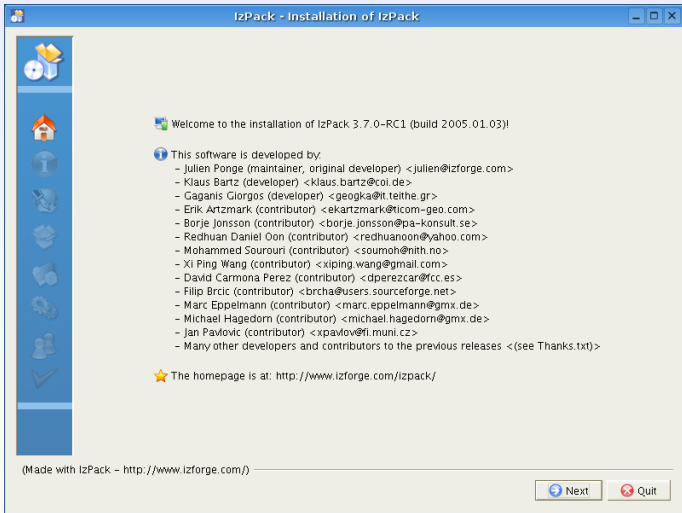


# Installation panels flow

- Panels define the installation steps. Several panels are available, some even do the same functional task. **You decide which panels you want and in which order.**
- For this tutorial, we will use the following panels:
  - ④ *HelloPanel*: welcome our user to the installation process
  - ② *HTMLInfoPanel*: display some informations with a structured text
  - ③ *LicencePanel*: the legal terms that must be agreed to reach the next installation steps
  - ④ *PacksPanel*: allow the user to pick the packs that she/he wants to install or not
  - ⑤ *TargetPanel*: choose where to install the files
  - ⑥ *InstallPanel*: performs the actual files installation
  - ⑦ *SimpleFinishPanel*: conclude the installation with a success.

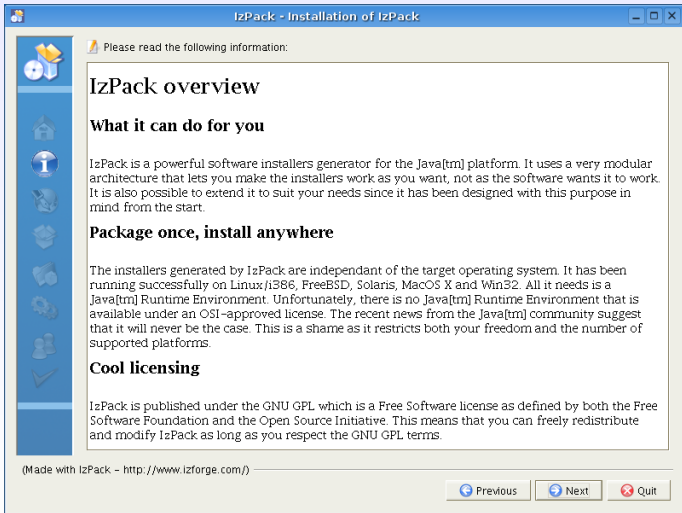
# Panels flow illustrated

## HelloPanel



# Panels flow illustrated

## HTMLInfoPanel



The screenshot shows a window titled "IzPack - Installation of IzPack". The main content area displays the following text:

Please read the following information:

### IzPack overview

#### What it can do for you

IzPack is a powerful software installers generator for the Java[tm] platform. It uses a very modular architecture that lets you make the installers work as you want, not as the software wants it to work. It is also possible to extend it to suit your needs since it has been designed with this purpose in mind from the start.

#### Package once, install anywhere

The installers generated by IzPack are independant of the target operating system. It has been running successfully on Linux/i386, FreeBSD, Solaris, MacOS X and Win32. All it needs is a Java[tm] Runtime Environment. Unfortunately, there is no Java[tm] Runtime Environment that is available under an OSI-approved license. The recent news from the Java[tm] community suggest that it will never be the case. This is a shame as it restricts both your freedom and the number of supported platforms.

#### Cool licensing

IzPack is published under the GNU GPL which is a Free Software license as defined by both the Free Software Foundation and the Open Source Initiative. This means that you can freely redistribute and modify IzPack as long as you respect the GNU GPL terms.

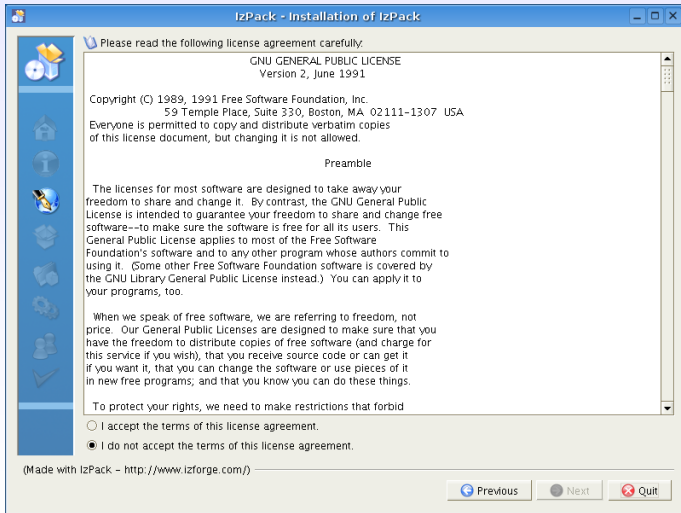
(Made with IzPack - <http://www.izforge.com/>)

Navigation buttons: Previous, Next, Quit



# Panels flow illustrated

## LicencePanel





# Panels flow illustrated

## PacksPanel (here *ImgPacksPanel*)

The screenshot shows the 'IzPack - Installation of IzPack' window. It features a sidebar with navigation icons, a main area for selecting packs, and a preview area for the selected pack.

**Select the packs you want to install:**

<input checked="" type="checkbox"/>	Core	4.51 MB
<input checked="" type="checkbox"/>	Documentation-HTML	0 bytes
<input checked="" type="checkbox"/>	Documentation-PDF	0 bytes
<input checked="" type="checkbox"/>	Documentation-NanoXML	434.42 KB
<input checked="" type="checkbox"/>	Sample	23.38 KB
<input checked="" type="checkbox"/>	Sources	3.2 MB

**Package snapshot:**

```
// Let's use the system LAF
ButtonFactory.useButtonIcons()
if (laf == null)
{
    if (!syskey.equals("mac"))
    {
        String syslaf = UIManager.
setLookAndFeel(s
ger.getLookAndFe
kAndFeel.setCurr
ctory.useHighlig
data.buttonsHCoLo
```

**Description:**  
The full IzPack source code.

**Note:** greyed out packs are required.

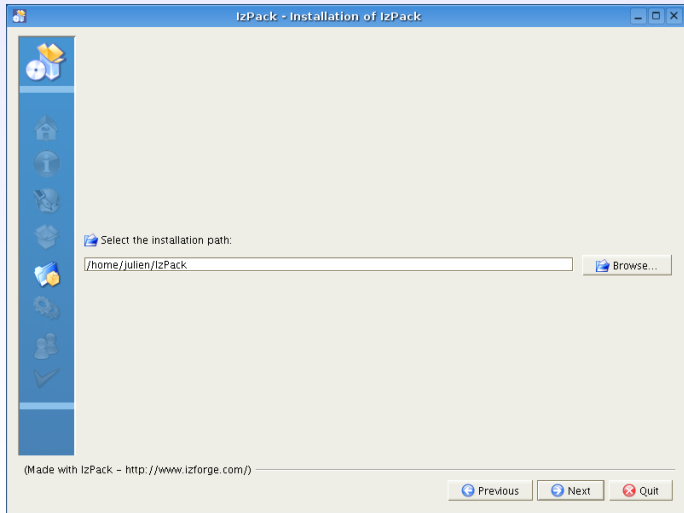
Total space Required: 8.16 MB  
Available space: > 2 GB

(Made with IzPack - <http://www.izforge.com/>)

Navigation buttons: Previous, Next, Quit

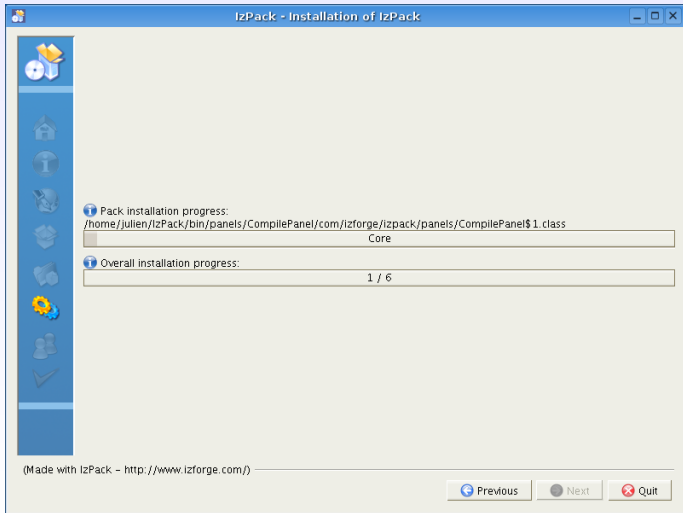
# Panels flow illustrated

## *TargetPanel*



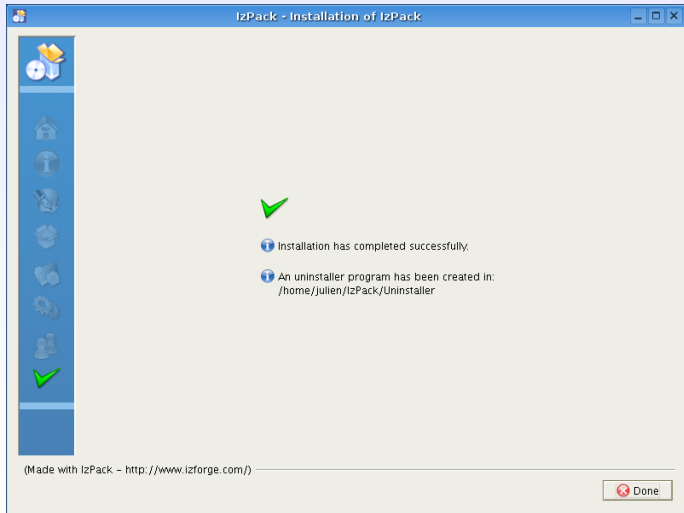
# Panels flow illustrated

## *InstallPanel*



# Panels flow illustrated

## *SimpleFinishPanel*



# Outline

- 1 Introduction
  - The IzPack project
  - Technical overview
- 2 **Making an installer**
  - Preliminary steps
  - **Creating the installation files**
  - Building the installer
- 3 Conclusion

# Basic installation XML file canvas

Here is a global view of what our file will look like:

```
MyApp-install.xml
```

```
<installation version="1.0">

  <info> (...) </info>
  <guiprefs (...)> (...) </guiprefs>
  <locale> (...) </locale>
  <resources> (...) </resources>
  <panels> (...) </panels>
  <packs> (...) </packs>

</installation>
```

# Global informations

We will specify here:

- the authors of the application to install
- the application name, version and url
- the minimum Java<sup>TM</sup> version required (optional).

## The *info* section

```
<info>
  <appname>MyApp</appname>
  <appversion>1.2.3</appversion>
  <authors>
    <author name="Snoopy" email="snoopy@myapp.org" />
    <author name="Foo Bar" email="foo@bar.org" />
  </authors>
  <url>http://www.myapp.org</url>
  <javaversion>1.4</javaversion>
</info>
```

# GUI tweakings

- We can customise the default size.
- We can specify a Look & Feel for a given OS, including:
  - Metal (hum hum ...)
  - Kunststoff, Metouia, Liquid
  - JGoodies variants.
- The default is to pick the native emulation L&F.

## The *guiprefs* section

```
<guiprefs height="600" resizable="yes" width="800">  
  <laf name="metouia">  
    <os family="unix" />  
  </laf>  
</guiprefs>
```



# Choosing the available languages

This step is quite easy. Just pick the ISO3 codes among the available languages, for instance:

## The *locale* section

```
<locale>
  <langpack iso3="eng"/>
  <langpack iso3="fra"/>
  <langpack iso3="deu"/>
  <langpack iso3="ita"/>
  <langpack iso3="jpn"/>
  <langpack iso3="spa"/>
</locale>
```

# Including the needed resources

- Each panel needs resources (see the IzPack documentation). A resource associates a path to a file and an identifier.
- Here, we have the following resources:
  - the text for *HTMLInfoPanel*
  - the legal terms for *LicencePanel*
  - an optional picture for the language selection box.

## The *resources* section

```
<resources>
  <res src="install-readme.html" id="HTMLInfoPanel.info"/>
  <res src="Licence.txt" id="LicencePanel.licence"/>
  <res src="langsel.jpg" id="installer.langsel.img"/>
</resources>
```

# Specifying the panels

Simply put the panels names in the obvious order:

## The *panels* section

```
<panels>
  <panel classname="HelloPanel"/>
  <panel classname="HTMLInfoPanel"/>
  <panel classname="LicencePanel"/>
  <panel classname="PacksPanel"/>
  <panel classname="TargetPanel"/>
  <panel classname="InstallPanel"/>
  <panel classname="SimpleFinishPanel"/>
</panels>
```

# Making the packs

- We use the smart Ant filesets syntax to pick the files contained in each pack.
- Target directories are specified. `$INSTALL_PATH` identifies the path chosen from *TargetPanel*.

## The *packs* section

```
<packs>
  <pack name="Core" required="yes">
    <description>MyApp core files.</description>
    <fileset dir="" targetdir="$INSTALL_PATH">
      <include name="*.txt" />
      <include name="bin/**/*" />
      <include name="lib/**/*" />
    </fileset>
  </pack>
  (...)
</packs>
```

# Outline

- 1 Introduction
  - The IzPack project
  - Technical overview
- 2 Making an installer
  - Preliminary steps
  - Creating the installation files
  - Building the installer
- 3 Conclusion

# Ant integration

- Ant is perfect for software tasks automation.
- IzPack can be integrated with Ant in a simple and elegant manner.
- IzPack needs the following informations (the same holds true for a command-line invocation):
  - the input installation XML file
  - the name of the output installer Jar file
  - the kind of installer (standard or web-based)
  - the base directory, to resolve the relative paths specified for the various files of the XML installation file (files, resources, ...)
  - the directory where IzPack is installed.

# Calling IzPack from Ant

## Warning

Make the IzPack compiler jar available in the classpath before you call Ant.

## Make the task available

```
<taskdef name="izpack" classpath="${izpack.dir}/lib/compiler.jar"
         classname="com.izforge.izpack.ant.IzPackTask"/>
```

## Call the IzPack task

```
<izpack input="MyApp-install.xml"
        output="${dist.dir}/MyApp-install-${ver}.${rel}.jar"
        installerType="standard" basedir="${dist.dir}"
        izPackDir="${izpack.dir}"/>
```

# Summary

- You have now made a simple installer for your application. You should now feel more confident with IzPack, however you can get much more from it.
- You can now try some more advanced features (such as desktop shortcuts generation or scripts token replacement) or play with some other panels as well.
- The following resources will be of a great help:
  - the IzPack documentation
  - the wiki at BerliOS
  - the mailing-lists archives
  - real-life examples (such as IzPack itself or open-source projects that use it).





# Supporting IzPack

If IzPack is useful to you and/or your company, please consider supporting it financially. Just think about how expensive are its proprietary competitors...

- I accept donations through PayPal with my email address [julien@izforge.com](mailto:julien@izforge.com).
- Missing a feature ? Then you can offer a bounty to an IzPack developer for implementing it. Ask for the feature and see if a developer wants to make it for a donation that you can negotiate at your own discretion with her/him.



# Credits

- The  $\LaTeX$  Beamer class.
- The BerliOS crew.
- The numerous past and present IzPack developers and contributors.

