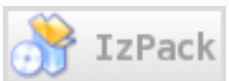
The IzPack tutorial

Getting started with a basic installer for your software.

Julien Ponge[1]

[1]<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

Outline

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...

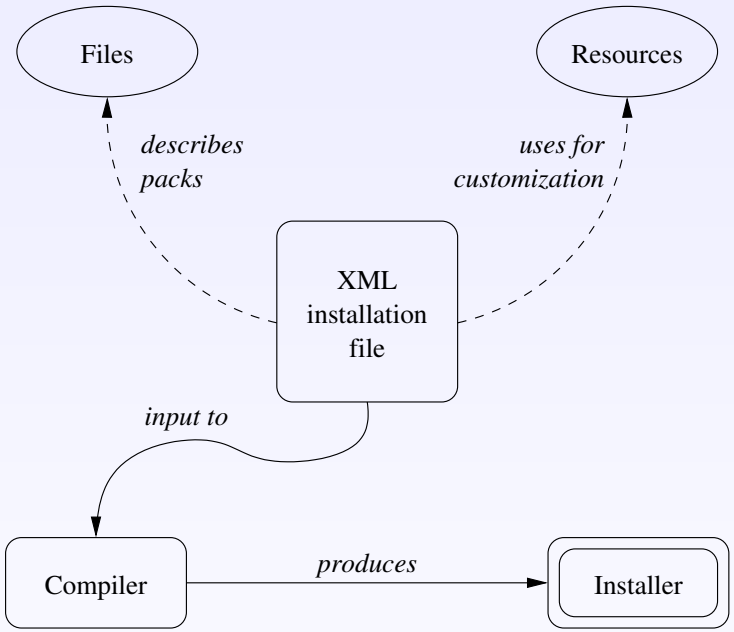
Project life

- 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

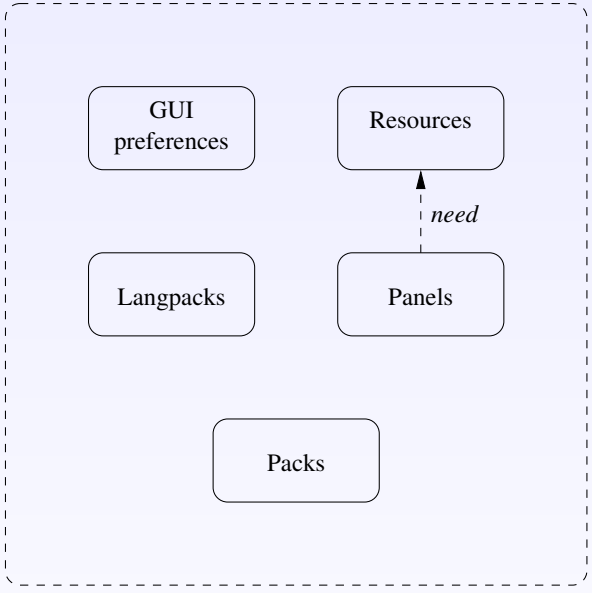
IzPack

- 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

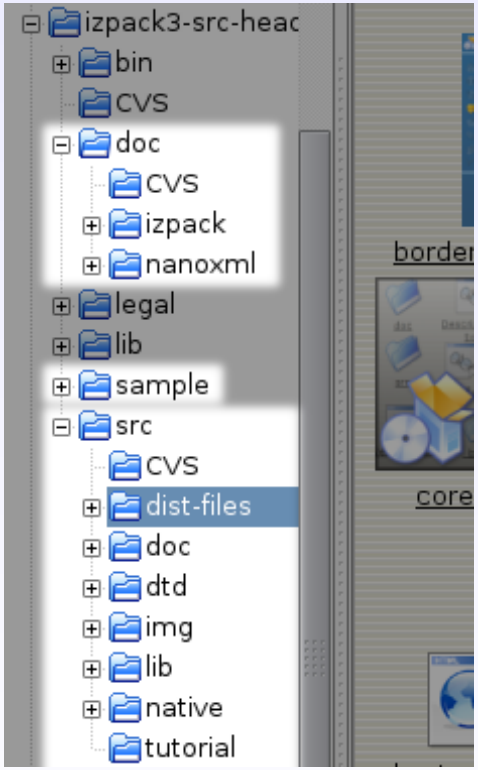
- An installer contains:
  - the real files in packs
  - the langpacks.
- An installer offers a set of panels that define the steps to perform an installation.
- Resources can be needed by panels and GUI preferences can change the look depending on the OS (size, L&F, ...).



Outline

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.

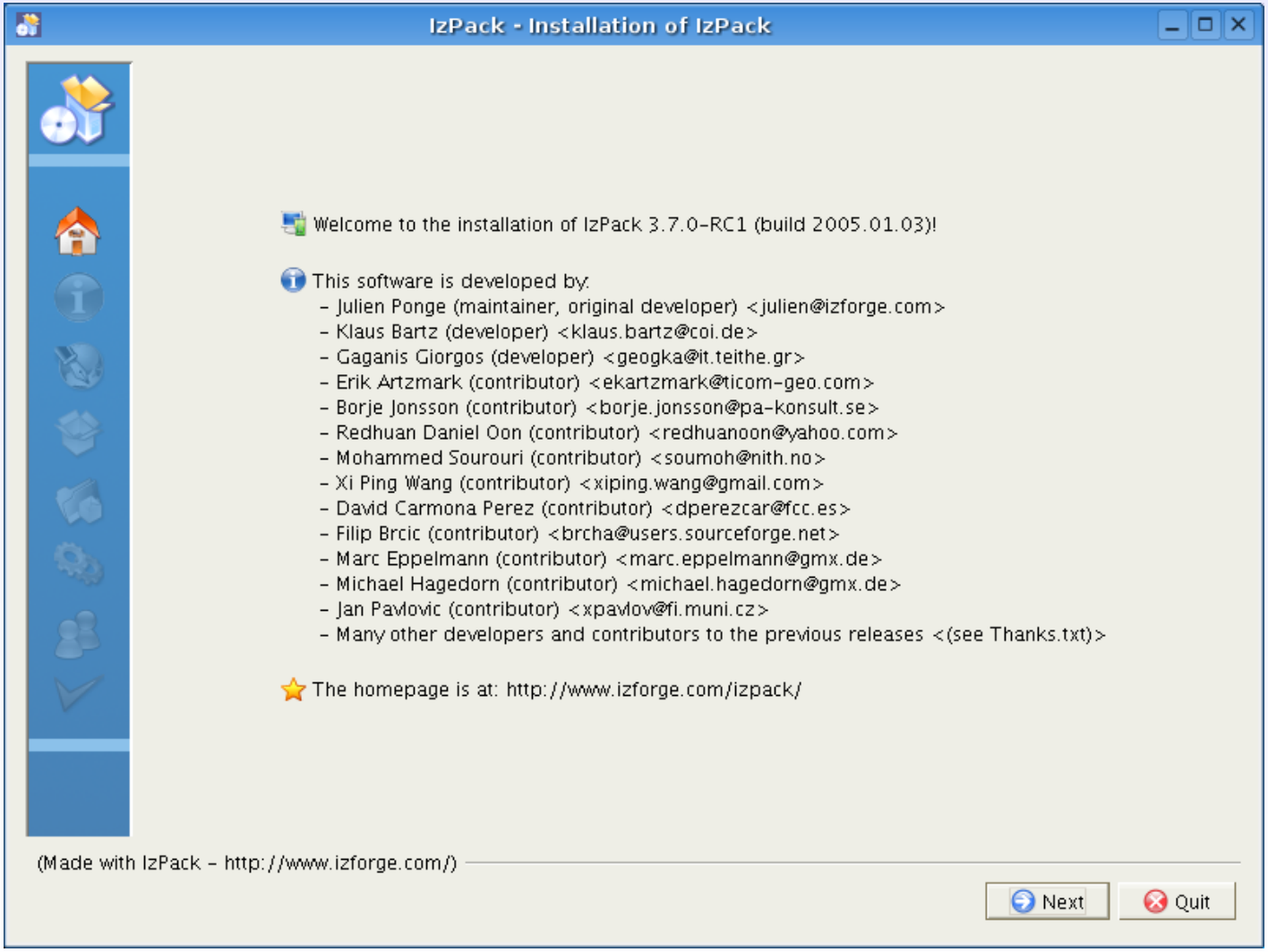


Installation panels flow

- 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:
  1. *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.

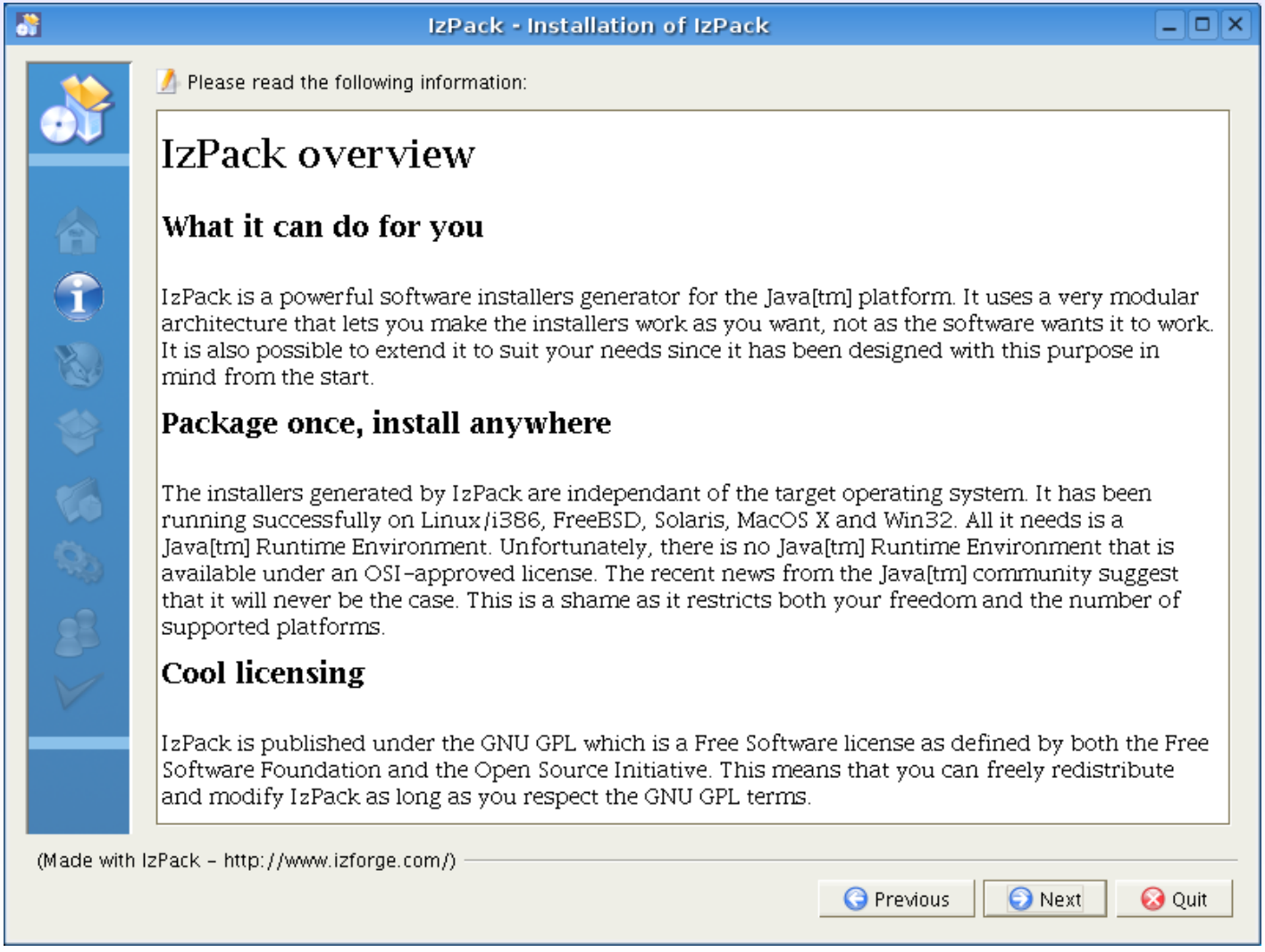
Panels flow illustrated

HelloPanel



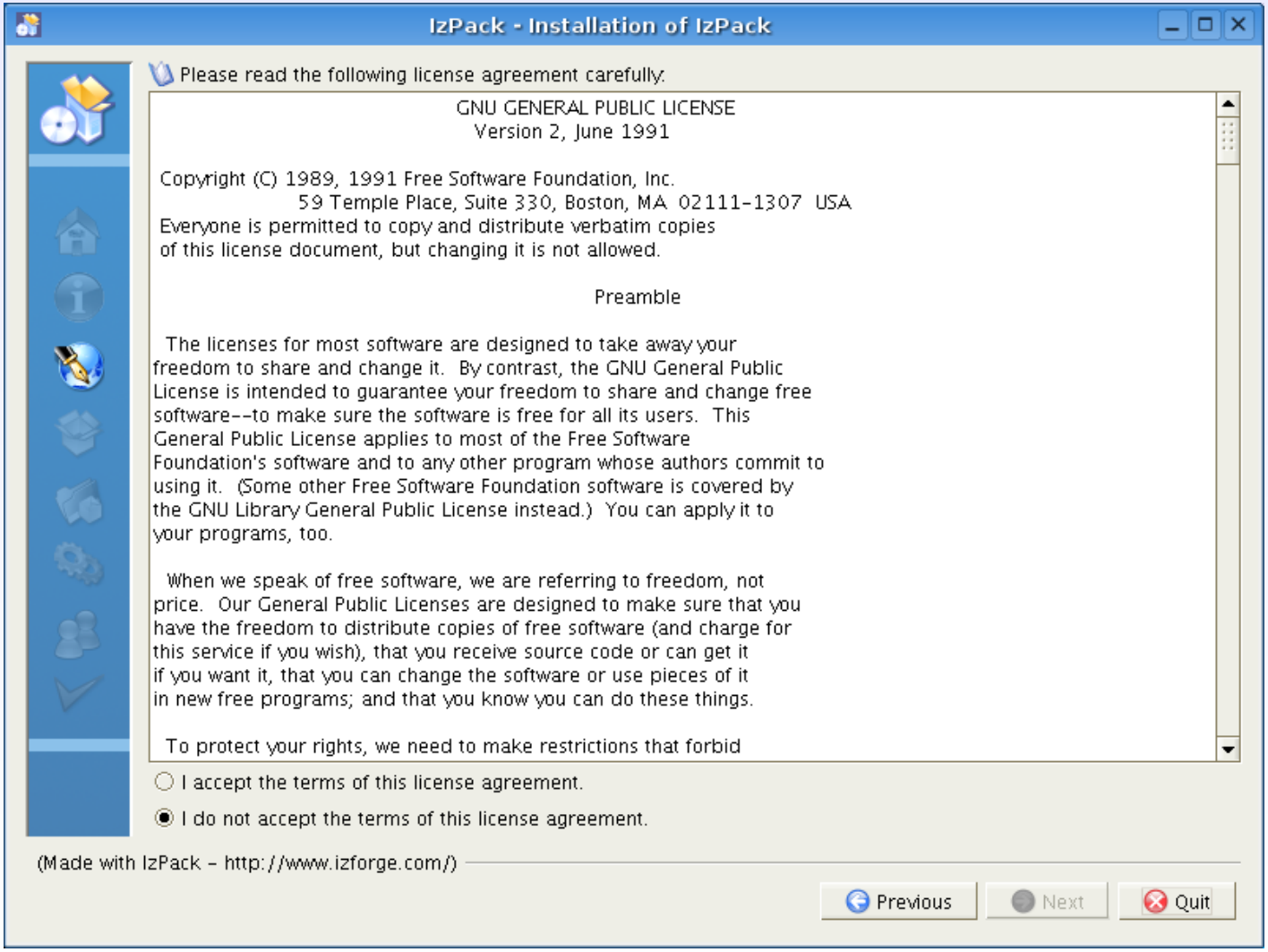
Panels flow illustrated

HTMLInfoPanel



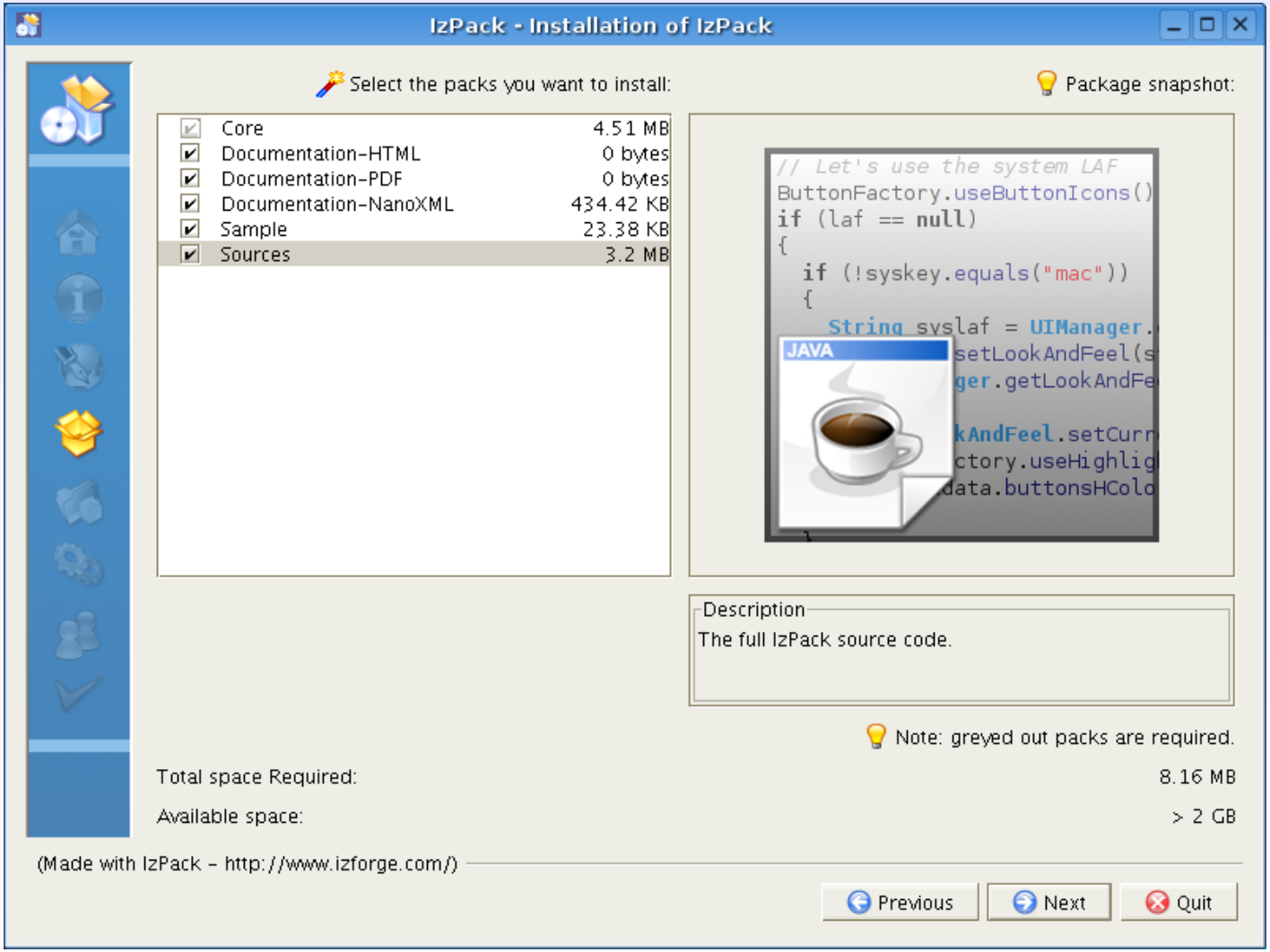
Panels flow illustrated

LicencePanel



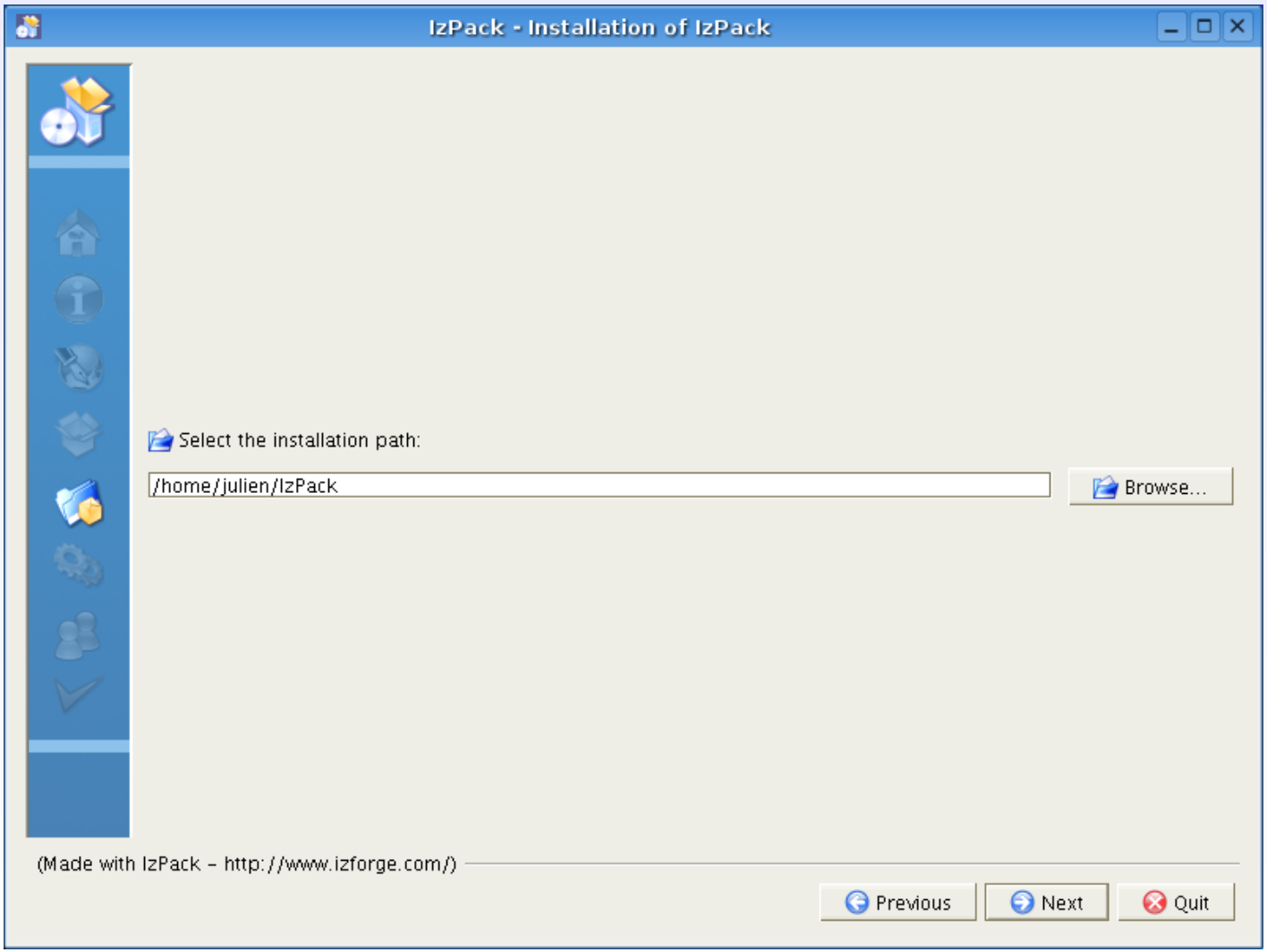
Panels flow illustrated

PacksPanel (here *ImgPacksPanel*)



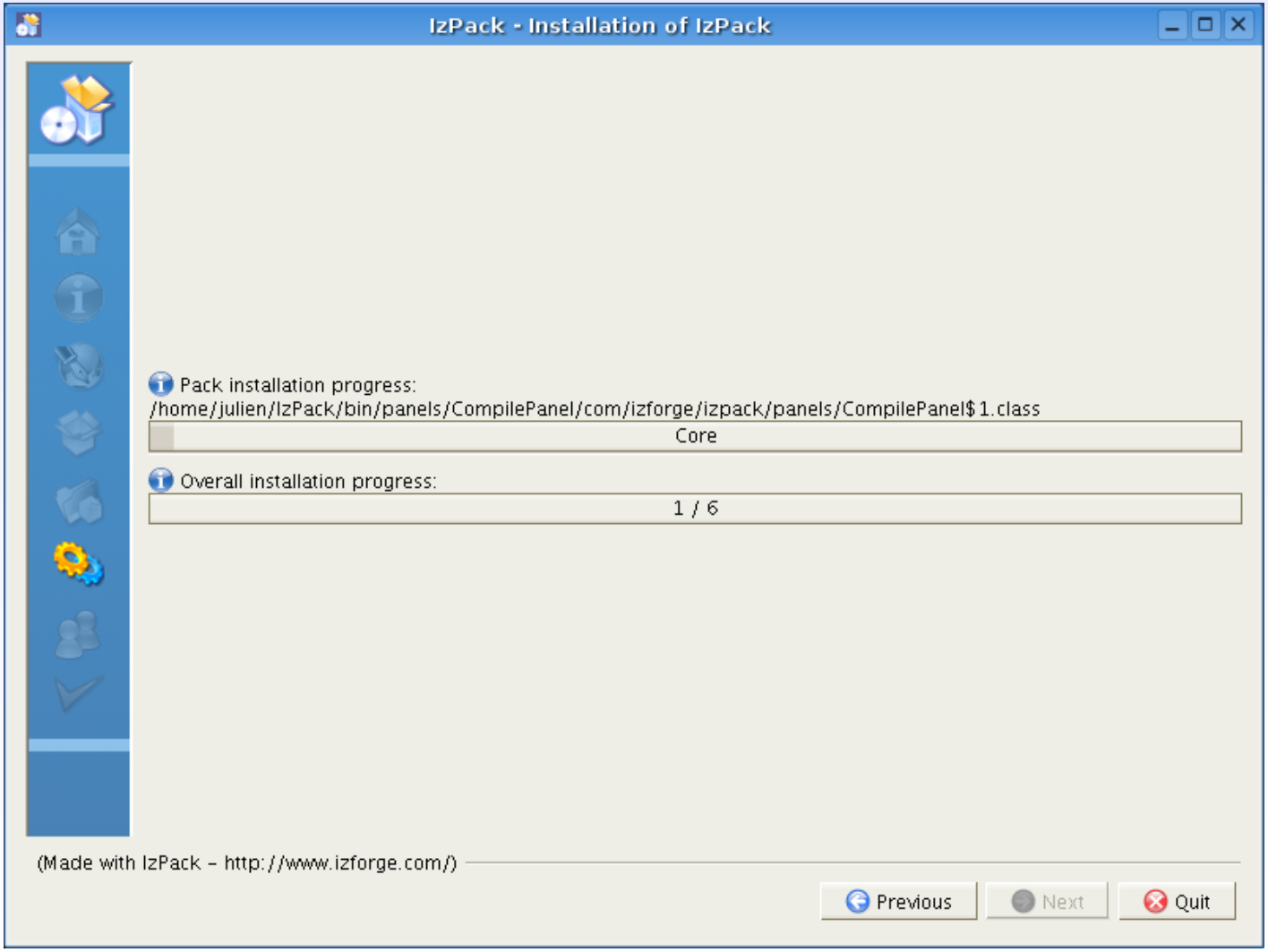
Panels flow illustrated

TargetPanel



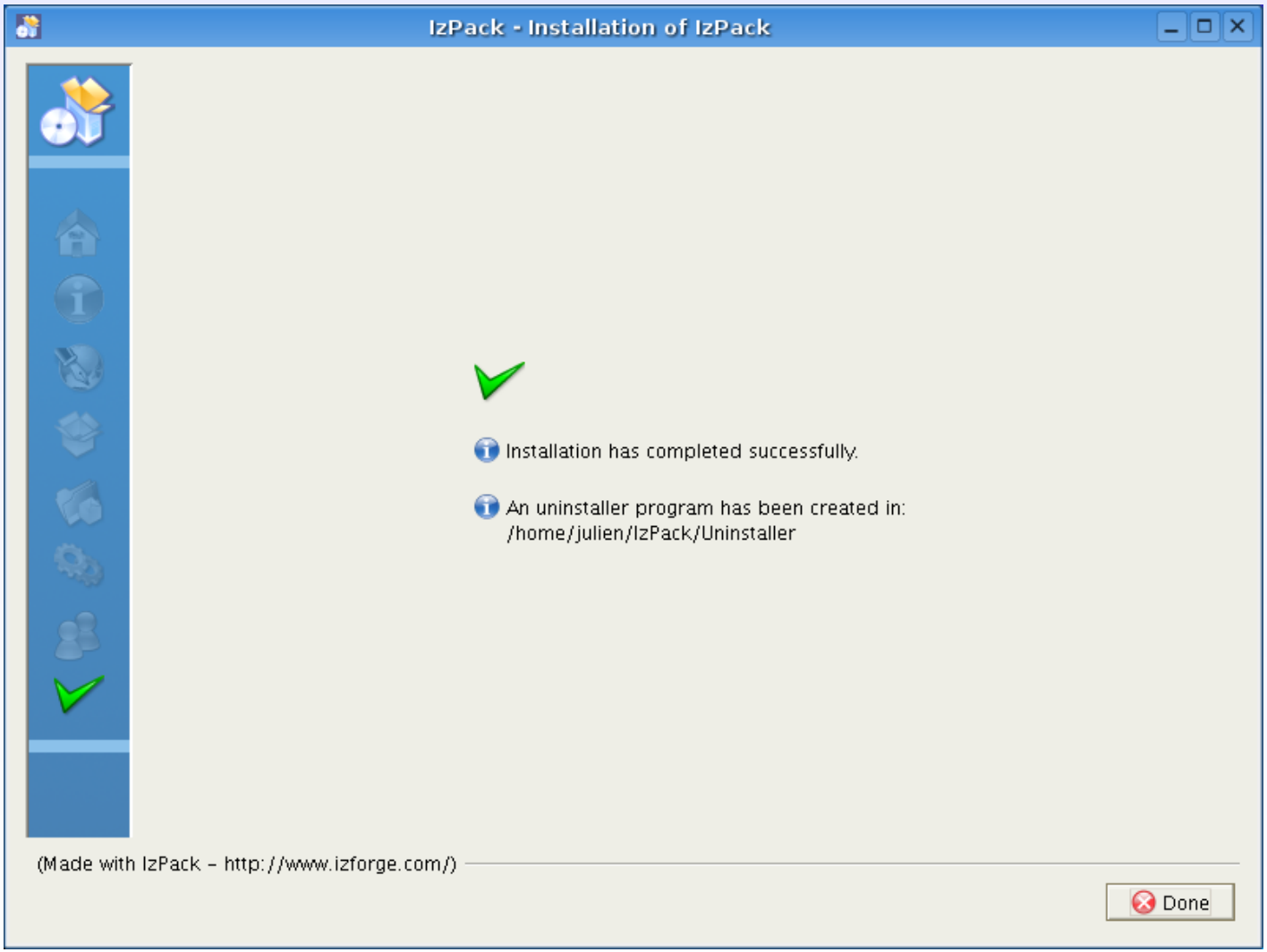
Panels flow illustrated

InstallPanel



Panels flow illustrated

SimpleFinishPanel



Outline

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™ 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

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`.
- 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.