IzPack Guidelines for Contributors.

Julien Ponge
<julien@izforge.com>
http://www.izforge.com/

November 2, 2002

Purpose of This Document

The purpose of this document is to present the guidelines that you should follow when contributing to IzPack. It should not restrict anyone willing to contribute, but rather help to keep things clean. Everyone has their own programming habits. However, when more than one person works on a project such as IzPack, it *is* necessary for all to work in the same way so that it does not mess up the whole project. What's more, new contributors must know how to work with the 'older' contributors. In particular, it is necessary to know the procedures to contribute.

How to contribute

Requirements

IzPack is a *very* open project. So it will be very easy for you to contribute. First of all, you need to join the IzPack mailing-list (see `http://www.izforge.com/izpack`). Secondly, you have to be familiar with CVS. We use CVS to manage the IzPack source code repository. It allows developers to work concurrently on the files, branch and re-merge the development tree and more. You can find the on-line manual and client software for CVS at `http://www.cvshome.org/`. Of course you can also get some help on the IzPack mailing-list.

You don't have to be a wizard to contribute to IzPack. Any contribution small or large is welcome. Remember, IzPack should be fun for the users and

developers :-) So join and be a part of it.

Getting a CVS Account

Getting a CVS account is not very hard. Just send e-mail to `julien@izforge.com` and explain why you need access. I will then create an account for you.

Before You Make Changes

Please first announce any changes you would like to make on the mailing list and respect the input from other members who have made contributions and helped shape the design of IzPack. Remember this is collective work, so please discuss what you want to do and how you want to do it. It is a good idea to reach agreement before starting to code, as this can save frustration later on.

Doing Your Work

Coding Conventions

The source code must be as consistent as possible, that's why :

- use a 2-characters indentation with *soft-tabs* (tabulations are replaced by spaces). This prevents us from badly laid out source code. Hard-coded tabulations can break the whole presentation, so **use** soft tabs.

- This is the preferred way of placing braces :

```
if (condition)
{
    // some nice code ...
}
```

  instead of

```
if (condition) {
  // some nice code ...
}
```

- Please comment your source code. You don't have to write a lot but you should describe how your code works and what the purpose of each class/function is. Since it is likely that others will have to maintain or interface with your code, possibly without your help, your comments will be much appreciated.

- If you have to add a new file, insert a header with the copyright notice as found in the other IzPack source files

Documenting Your Work

When you make changes such as modifications or additions to the infrastructure or you add a new feature, then you should write additional documentation.

Infrastructure Work

If you make infrastructure changes or enhancements, please write useful developer documentation. The important point is that those who make additions after you, use your infrastructure modifications correctly. Since the documentation that comes with IzPack is often the only means of learning how everything works, this becomes very important. Obviously, doing so will also enhance the value and recognition of your work.

In many cases it should be sufficient to provide your description in the class header (javadoc style documentation). This is particularly true when the implementation is limited to a single class. In cases where an entire package has been added you should write a javadoc addition that describes the entire package. To do this simply place a HTML file in the package directory that has the name `package.html`. Then place your HTML content in between the tags:

```
<body>
my HTML content goes here...
</body>
```

Don't use the `<HTML>` tag, as the content of this file will be inserted as is into the automatically generated javadoc. If needed, more extensive documentation can be placed in a sub-directory called `doc-files`. It is also possible to insert pictures and diagrams by using HTML tags, if your work should require this. You can see an example of how this is done in `com.izforge.izpack.util`. For more details see the Java tools documentation that comes with the documentation package from Javasoft.

In simple terms: please leave the same kind of documentation behind when you are done with your work, as you would enjoy to find when you start :o)

While it is important to write documentation, please also make use of the documentation that is provided to inform yourself about the IzPack infrastructure before you plan any additions or modifications.

New Features

If you add new features, I am sure you want them to be used extensively. The best way of ensuring that this happens it to document them well for the users of IzPack. Small and especially hidden documentation gets often overlooked and so does the feature that it describes. Also, incomplete, incorrect or misleading documentation tends to make it hard on users to use a feature. As a result, if it is not vitally important, most just don't use it at all.

It is not necessary to add your documentation directly to the user manual source. You can send me a plain text file and include pictures and diagrams if they should be required. It is also not necessary that everything is perfectly described in the best English. Often, someone on the mailing list can help out to get the text into publishable form. Just remember that the resulting information in the manual might be in better English but it can only be as good and as complete as the information that you provide.

Committing Your Changes

Test your changes with the latest code version from the CVS repository. When it works, you can commit it to the main CVS branch. Sometimes though, you might be asked to use a new branch for your changes, and we will merge it later. If you make big changes, you should also create a specific branch and tell the others that and why you need it. Don't forget to write a brief comment when committing your files. After the commit, please announce on the mailing-list what you did and don't forget to explain how to

use your code/feature.

Don't Be Shy!

Some people tend to worry about their English on the mailing-list. Firstly we aren't here to find out who speaks the best English. Secondly, these people usually speak better English than they think! So don't be shy and express yourself freely :-) We all don't usually worry too much about it.