

Re: Giving an application a window icon in a sensible way

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-11/msg02426.html>

- *From:* "Twisted" <twisted0n3@xxxxxxxxxx>
 - *Date:* 19 Nov 2006 20:00:39 -0800
-

Mark Rafn wrote:

Twisted <twisted0n3@xxxxxxxxxx> wrote:

I want to give a Java application a window icon in a manner that is independent of how it is installed, etc.
The trick is obtaining an Image object for `myJFrame.setIconImage()`. Loading it from a URL means it won't work without a working network connection, and I need to host the image somewhere.

Not at all. A URL doesn't always mean http. It could be a file URL, or a resource URL inside your jarfile.

File URL and jar I was considering separately.

```
URL imgURL = getClass().getClassLoader().getResource("img/path");
```

Works if the image is in a jarfile OR a directory on the classpath.

Hrm.

Anyway I found something interesting. My google-fu isn't as weak as I thought — I was eventually able to dredge up a way to encode icons into a class file.

It involved exporting the file from photoshop as an XPM, pasting most of the result into the declaration of a string array, and feeding it to a class named `XImageSource`. Of course, this turned out not to be a standard library class, and tracking it down posed its own challenge (during which time Firefox crashed for the second time today — it hit a crapplet on a page somewhere and died the number. It actually tottered along sort-of-working until I quit it, but wouldn't load anything — and after being quit I couldn't start a new instance until I terminated a bunch of firefox-related processes that were idling in

Re: Giving an application a window icon in a sensible way

the task manager that didn't have any UI or cpu activity!).

Naturally, the XImageSource class had dependencies to track down as well.

Naturally, one of those dependencies had a bug — XpmParser. It had

```
colors = new int[charsPerPixel*2];
```

where it appeared to need

```
colors = new int[(charsPerPixel == 2)?65536:256];
```

since it actually multiplies one char by 256 and adds a second in the latter case, and was throwing ArrayOutOfBoundsException like they were going out of style.

Naturally, the author of those classes included a copyright notice and will probably sue me for copyright infringement for fixing their bug without permission, too, now that I've copped to this heinous act in a public newsgroup posting.

But it actually works, and the source code is completely self-contained, without requiring any extra files besides the .java files. Which is what I was hoping to accomplish.

Thanks anyway. :)

.