Now Wait a Second:

Adrian Kosmaczewski

2007-01-27

There was a time where programming in Java was funnier than it is today.

Back in 1997 I published in my own web page my first Java applet. It was a calculator. That was way cool; I was able to share with the visitors of my site a whole program: not only the source code, but the whole thing, working, available for anyone to use, no matter the browser, no matter the platform, no matter whe geographical location.

You can play with that applet here. It still works on my MacBook and on my G5, almost 10 years later, running under the Java 1.5 virtual machine. I am simply amazed (It might not be the most interesting code you'll ever see, nor the most beautifully architectured one, but it did what I wanted it to do… so I consider it a success. Thankfully I see things different now).

So there was a time were Java was cool, new, easy to learn, the way to go. And here's a story to show how cool it was.

In 1998 I was in Buenos Aires, working in a dot-com company, programming and designing a website; we were using Windows NT 4 servers, and as such the web server was IIS, and the whole thing consisted of "classic" ASP pages, programmed using VBScript. Talk about technology; Java was non-existent in this environment. However, it was for me the basis for a small solution that proved really useful, during many years.

Our CEO came one day with an idea; in our website forum, the users should be able to upload images showing the products being offered. The images were uploaded on the server, and then they were showed beside the text of the offer. Rather simple requirements, right? Well it turns out that those images could have any size; I do not mean in bytes, but in pixels. Said like this, it does not seem like a big deal, but actually this poses a basic design problem. Since the width and height of the images are arbitrary, placing them correctly on the web page, with the right proportions, brought some problems to the technical team (that was just 2 people: my colleague and I). And so we realized that being able to know the width and height of the uploaded images was the missing link to solve the problem (and to keep our CEO happy).

I started to look for external components fulfilling the task, but without success.

Apparently there were no available components with the required feature: being able to return the width and height of images uploaded to the server. As such, I decided that this was an interesting thing to tackle, and I decided to create such a component by myself.

That's the "advocating inventor" thing that you can read in my Personal DNA profile. I love to do this kind of investigations.

First, since PNG files were not (yet) handled correctly by the mainstream browsers of the time (Internet Explorer 4 and Netscape Communicator 4), we decided that supporting GIF and JPEG files was more than enough for our needs. The challenge, then, was to parse the files looking for that information.

The first thing to do, is detect the file format. And for that, you just can't trust the file extension. A malicious user could upload a ZIP, an EXE or a DOC file, with a misleading extension, and your component would fail. The idea then is to open the file, regardless of the extension, and find a way to guess the file format reading the bytes inside. This is usually easy, since the first three bytes of a GIF file are numbers 71, 73 and 70: the letters, G, I and F, as you might guess. JPEG files start with the numbers 255 and 216. Easy to read.

Once you have the format, just choose the right algorithm for getting the width and height information. For GIF files, this is not a big deal: the width is stored in bytes 6 and 7, and the height in bytes 8 and 9. This explains why the maximum width and height of a GIF file is 65,535 pixels, since 65,535 = 216 - 1. Which makes for a really big image anyway.

But the tricky part was yet to come; JPEG files have a really, really weird way to store the width and height. It is just not stored in a fixed place, like in the case of GIF files; the position of this information is stored in the file, somewhere, and then, using that information, you must "jump" to that part of the file. Anyway, this is how the algorithm goes:

- Scan the file, from the beginning, three bytes at a time, until you find a sequence like this: 255, 195 and 192 (yes, it's weird)
- Once you find that sequence, stop searching; from the byte that contains 192 (with address "x"), the height is stored in bytes x + 5 and x + 6, and the width in bytes x + 7 and x + 8.

As you can imagine, when I found the algorithm, after a couple of days searching for it, I was extremely happy! I must thank the guys from the comp.graphics.algorithms newsgroup, who pointed me to the right resources at that time.

Looking back in time, I asked myself, why did I do it in Java? Well, first of all, because it was a fine and strict programming language to work with, that I had in my work computer and at home, and in which I could create a stable implementation, in a language easy to read. The environment dealt with memory management for me, was stable, free, and the end result ran quite fast in every computer I tested it.

Of course, our web server being a Windows box, and not having a Visual J++ license at work, we decided to rewrite it in Visual Basic 5, and create a COM component with it. This way, we could install it in our server, and it ran gracefully for more than 2 years before being replaced.

Later on, since the code did not handle the new JPEG 2000 format, we came accross some files that could not be handled properly by the component. Nevertheless, it proved useful and handled, almost without flaw, during 2 years, 24/7, lots of GIF and JPEG files that our customers uploaded to the server. Maybe it failed to recognize a couple of files, over tens of thousands, in a live environment.

Actually, now that I think of, I realize that at the time we completely overlooked the fact of the ominous GIF licensing thing of Unisys, but since

> only the software firms who sell the enabling software for profit would be expected to secure a licensing agreement from Unisys.

I think that we were outside of the problem. Besides, their patent rights are no longer what they used to be:

> The Unisys patent expired on 20 June 2003 in the USA, in Europe it expired on 18 June 2004, in Japan the patent expired on 20 June 2004 and in Canada it expired on 7 July 2004. The U.S. IBM patent expired 11 August 2006, The Software Freedom Law Center says that after 1 October 2006, there will be no significant patent claims interfering with employment of the GIF format.

(Source: http://www.gnu.org/philosophy/gif.html)

And now, you can download the code from my imgsize project page. Feel free to play with it! As usual, do it at your own risk. All comments welcome, of course! And I hope that nobody will get sued for this, either :)