*Astronomical Data Analysis Software and Systems IX*
*ASP Conference Series, Vol. 216, 2000*
*N. Manset, C. Veillet, and D. Crabtree, eds.*

# The Aladin Java Applet Experience

P. Fernique, F. Bonnarel

*Centre de Données astronomiques de Strasbourg.*

**Abstract.** The applet feature has certainly been the reason of the success of the Java language fast development: with a simple mouse-click it became possible to download remote executables and to run them via standard browsers such as Netscape or IExplorer. However, three years after its first release, many Java developments are in fact standalone applications which require an installation of the JVM (Java virtual machine) rather than applets. In which conditions is the applet feature a really applicable concept? This paper presents the CDS (Centre de Données astronomiques de Strasbourg) experience of Java applets through the development of Aladin Java. We present the limitations that we encountered, and how to overcome them: support of all browser versions, insuring sufficient performances, dealing with the access restrictions to local disks and in particular keeping the code small enough to be easily downloaded.

## 1. The Aladin Project

The Aladin project goal is to help to the cross identification of astronomical objects. To do that we have implemented a software displaying Schmidt plate scans on which we can superimpose catalogs and database field charts. The image scans come from the MAMA machine at CAI (Guibert 1992), and from the STScI for the DSS1 and the DSS2 (Lasker 1994). The overlays come from the Simbad database (Wenger et al. 2000), the VizieR server (Ochsenbein 2000) with its two thousand catalogs and the NED database (Madore et al. 1992).

After a first release using the X11 technology, we decided to build a Java prototype to implement the Aladin user interface as *a Java applet*. Two years ago, the applet concept seemed to be a good technical solution to solve two problems that we had encountered with the X11 implementation:

- The installation cost requiring user availability and competences;
- The portability cost requiring engineer time and efforts.

## 2. The Java Applet Concept

A Java program is compiled only once on the development machine. The executable code, called P-code, is downloadable and runnable directly by most current browsers. To run the P-code the browsers integrate a Java virtual machine, called JVM, which is able to execute the P-code. Consequently, the Java applets don't require any installation and all platforms supporting a Java enabled browser run them immediately.

## 3. The Four Applet Surprises

These features have certainly been the reason of the Java language success. However, there are presently relatively few applets on the net. In fact, the Java softwares are very often distributed as standalone applications requiring not only their own installation but also the Java virtual machine installation.

This fact can be explained by these four typical applet user questions: *Why is it so slow to start up? Why is it so slow to run? I cannot print ! I cannot use local data ! It works on Netscape but not on IExplorer !* Behind theses four reactions, there are four applet problems.

### 3.1. The Network Downloading Delay

Each time a user wants to run an applet, its P-code has to be downloaded through the network. Nowadays, a software having a few megabytes is quite common, but for a Java applet, with the present internet bandwidth, it is strongly recommended to limit its size to a few hundreds of kilobytes.

It means concretely that making use of the additional packages (3D-image manager, XML parser, ...) is not recommended in an applet because such applets are generally too big to be downloaded.

### 3.2. The JVM Performances

A good JVM implementation is generally 2 to 5 slower than the C equivalent code. But now, another aspect of the JVM performances appears with the new 1.2 Java release: some foundation classes, which come with the Java language toolkit, can be slower in the more recent release. The main reason is that these classes offer more and more sophisticated features. But there are not always required by the application and often heavily resource consuming. We have measured that the same Java code is 10% slower between Java 1.1.4 and 1.1.8, and 34% slower between Java 1.1.8 and 1.2.2

### 3.3. The Security Restrictions

By default, Java applets cannot access to the user machine resources. The main consequences are: no local disk access, no printer access. In addition the network access is restricted to a dialog with the applet server i.e. the machine which sent the applet code to the user machine. The new concept of "secure applet" allows one to remove these restrictions, but practically, it is no easy to implement this feature (Netscape and IExplorer have different standards) and only the most recent browsers support it.

### 3.4. The JVM Incompatibilities

Java is claimed to be always compatible... but there are two major problems:

- The first incompatibility problem is about the configurations. For Aladin, we have counted 103 different machine configurations - i.e. combinations of different Java releases, different hardwares, different operating systems and different browsers. If a bug can be harmless on one configuration, it will certainly jam others. The only solution is to test on each configuration;

- The second incompatibility problem is about the Java machine itself: there exists three major Java releases. So if an applet uses one package object of the last Java version, it will be incompatible with the previous releases. The table below shows the JVM release of current browsers. An important point is that the 4 years old first Java release is still used by 8% of web users and the last release has not yet been integrated in the browsers.

| Release | Browsers | Packages | Aladin java current usage |
|---|---|---|---|
| 1.02 | NS2, NS3, IE3 | 8 | 8% |
| 1.1.x | NS4, IE4, IE5 | 20 | 87% |
| 1.2.x | - | 50 | 0% |

A first solution to overcome these problems is to ask to the users to update their browsers according to the java release of the applet that they want to execute. But in doing this, we loose the applet concept and, as the table shows, it seems that the users are not updating their browsers - most likely they just change their browser when they replace their computer.

The second solution is to use only the compatible Java features. It means that if a developer wants to allow 90% of Web users run its applet, he has to restrict his Java development to the 20 Java 1.1 packages. And for example, if the developer uses the Swing package, he should be aware that two years delay and perhaps more will be required to reach the majority of the web users. This fact is certainly the most unpleasant problem with the Java applets.

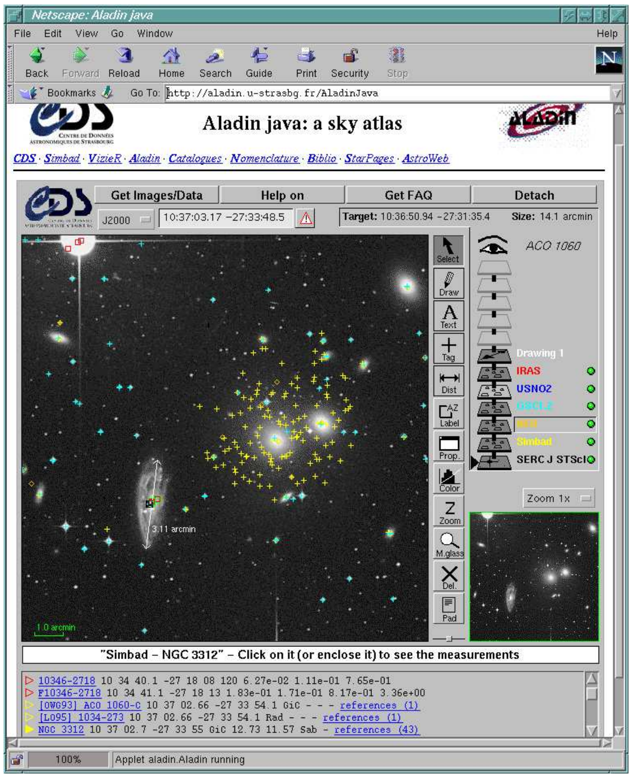
## 4. The Aladin Applet Technical Choice

In spite of these applet constraints, the first release of Aladin Java has been accessible since January 1999. Presently, this applet has the following characteristics:

- It is 170kB large and avoids any additional package;
- It doesn't integrate too sophisticated Java classes in order to keep good performance. In this sense, the class for image manipulations or for data bufferisation has been rewritten just to implement the features needed by Aladin;
- It has been necessary to implement on the applet server a "proxy" mechanism: when the Aladin applet needs an access to an other remote machine, for instance the Simbad machine, it asks the applet server to query the Simbad machine for him, and to send the result back to it. The actual proxy installation is based on the GLU system (Fernique et al. 1997);
- Aladin is Java 1.02 compatible and uses only the 8 main packages. It was tested on the most frequent platforms: PC (Windows + Linux), Sun (Solaris) with Netscape and Explorer since the release 3. These configurations cover 91% of our web users.

## 5. Conclusions

As we have shown, developing true Java applets that can be executed by every Web user implies difficult choices from the developer's point of view; but if he is




ready to do them, he can take all the benefit of this great concept. Aladin applet has been accessible by adding just a new link in our home page (http://aladin.u-strasbg.fr/AladinJava), and now we have more that 200 queries per day (images + overlays) and only 2 user questions per month.

## References


Bonnarel, F. et al. 1999, in ASP Conf. Ser., Vol. 172, Astronomical Data Analysis Software and Systems VIII, ed. David M. Mehringer, Raymond L. Plante, & Douglas A. Roberts (San Francisco: ASP), 229

Fernique, P. et al 1997, AAS, 191, 1705

Guibert, J. 1992, in Proceedings of the Conference on “Digitised Optical Sky Surveys”, eds. H. T. MacGillivray & E. B. Thomson (Kluwer Academic Publ.), 103

Lasker, B. M. 1994, in IAU Symposium 161, Astronomy from Wide–Field Imaging, eds. H. T. MacGillivray & E. B. Thomson (Kluwer Academic Publ.), 167

Madore, B, et al. 1992, in ASP Conf. Ser., Vol. 25, Astronomical Data Analysis Software and Systems I, ed. D. M. Worrall, C. Biemesderfer, & J. Barnes (San Francisco: ASP), 47

Ochsenbein, F. 2000, A&AS, to be published

Wenger, M. et al. 2000, A&AS, to be published