Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

Java: The Application and Data Distribution Vector for Astronomy

A. Micol, R. Albrecht

ST-ECF, ESA/SSD/SA, ESO

B. Pirenne

ST-ECF, ESO

Abstract. Among the tasks of the Space Telescope European Coordinating Facility (ST-ECF) is providing to the astronomical community software for the retrieval, calibration, and analysis of HST data. This has been plagued by notorious problems of programming language incompatibility, platform incompatibility, installation problems, on-line help/documentation and user interface inconsistencies, etc.

The combination of Java with state-of-the-art network browsers has the potential to solve many of these problems. In addition, it eliminates potentially paralyzing throughput problems by pushing the processing load to the client machine. This is important for post-retrieval re-calibration of multiple data sets from science data archives.

First experiments were conducted in the area of NICMOS support tools, and for the preview of data from the HST archive. Examples are shown, and experiences are discussed.

1. Introduction

One of the tasks of the Space Telescope European Coordinating Facility (ST-ECF) is to provide to the astronomical community software for the calibration and analysis of HST data. From long experience with such attempts, we find that the following problems exist:

- Programming language compatibility (different languages/compilers, etc.).
- Installation problems.
- On-line help/documentation.
- User interface (incompatible window managers or operating systems).
- Platform incompatibility.
- Remote maintenance (the need to notify/distribute a new version).

There have been many attempts to solve the above problems, some more successful than others. A successful approach has been to use a *data analysis host system* of the MIDAS or IRAF type which, if installed properly on the target site, will overcome most, if not all, of the above mentioned problems, except for the remote maintenance. The main disadvantage of this approach, however, is that it cannot benefit at all from software development going on in other disciplines, or in the commercial sector: with the exception of IDL, none of the major astronomical data analysis systems is being used outside of astronomy.

2. Java on the Web

Astronomers were quick to recognize the advantages of the World Wide Web for the distribution of information and documentation, and the Web has had a tremendous impact on astronomy.

While the distribution of information (text, images, video, sound) was thus solved, there still remained the problem of distributing what might be called computational functionality. Workarounds were introduced: a CGI (Common Gateway Interface) compliant software program was invoked on the server via a fill-in form, and the result was sent back as a Web page.

We note that this approach already solves many of the problems listed in the previous section: all users will be able to invoke this service as long as they have, for their particular platform, a Web browser which supports HTML forms. The problem of software system versions is also tackled, since there is now only one version: the one on the server. However, the most serious obstacles—that invoking the service puts a computational load on the server, and the stateless nature of the HTTP protocol—are still present.

In 1995, Sun Microsystems released Java. Its main advantage is that it produces architecture-neutral compiled code (applet) which can be transferred across the Web and can be statefully executed within the Web browser. If a link to such a page is followed, the page (encoded in ASCII) and the applet (encoded in ASCII-like pseudo-binary, called bytecode) are transferred to the client. The page contains the information required to operate the software, and the object code executes within the Web browser of the client, using it as a virtual machine.

The advantages of this approach cannot be overstated:

- No programming language incompatibilities (no re-compilation).
- Installation problems are eliminated: the software is ready to execute as soon as it arrives across the net.
- On-line help and documentation can seamlessly be provided through the hypertext capabilities of the Web.
- HTML allows easy user interface creation. In addition, the Web browsers provide a good UIF paradigm.
- Platform incompatibility is eliminated. Any platform which supports a Java capable browser becomes usable. The penetration of the Internet by the Web will make sure that the number of supported platforms will grow, and the capabilities of the browsers will be increased (since the commercial sector is very interested in the public having easy and cheap access to the Web).

- Remote maintenance. Any first link traversal and any reload forces a transfer of the current, i.e., most recent version of the page, plus the applet. This means that any change made to the software by the original authors is propagated to the users in a rapid and painless manner.

3. First Experiments

When the above considerations became obvious in late 1995, we started a process of familiarization with Java. This was made difficult by the fact that Java was a moving target at that time: even books on the subject printed as late as October 1995 were already partially obsolete. And, we only had a beta release of the Java capable Netscape browser.

After overcoming these initial difficulties, however, it was possible to implement a few test applications in a relatively painless manner. In fact, we found that for many applications which do not require the software to run continuously, it was advantageous to use Javascript rather than Java applets: the code is more readable than the code for an applet, and it is visible on the HTML level, so that users can convince themselves of the correctness of the computation. They also can save the code locally and enhance or modify it.

This approach is particularly appealing for incorporating applets into technical documents, which normally only provide formulæ and, at best, examples. The Instrument Handbooks for the science instruments of the Hubble Space Telescope are a good example for this: they contain numerous formulæ and algorithms to calculate or estimate integration times, signal-to-noise, background, unit conversion, etc. We intend to produce applets for many of these functions, and to provide links to them from the electronic version of the handbooks.

3.1. Magnitude/Jansky Converter

As a working example, we developed the Magnitude/Flux/Jansky Converter.[1] The tool is meant to help the community during proposal preparation for the NICMOS (Near Infrared Camera and Multi-Object Spectrometer) instrument, installed in the Hubble Space Telescope (HST) during the 2nd Servicing Mission in February 1997.

This tool is based on a fill-in form handled by a Javascript. The Javascript can recognize and respond to user events to behave like a spread-sheet: as soon as an input field is changed, all the others are automatically re-computed according to the chosen units. An error message is displayed if the input value is not appropriate. This feature, in particular, is not implementable with the CGI approach, where the form has to be submitted before the actual verification can be performed.

3.2. Archive Data Preview

For several years, the HST archive has allowed previewing the data before requesting the retrieval of the entire data set. For images, this has been little more than a static display, but for spectra, the preview facility included a set of quick-look analysis tools like zooming and examining actual pixel values. As

[1] http://ecf.hq.eso.org/nicmos/tools/nicmos_units_tool.html

these relied on the availability of certain utility software at the remote site, those tools needed to be distributed together with the user interface.

With a Java applet adapted from a template found on Web, it was possible to prepare a very useful—if primitive—tool to visualize our spectra and provide some data visualisation aid. With a simple mouse click, the software is loaded and starts to run. The first thing it does is to load the data from the ST-ECF database. The data are then displayed in a graphical form within the Web page. The software then sets itself into a mode allowing user interaction: using the mouse and keyboard keys, basic functions such as zooming and pointing are available. All the interaction and display is obviously done on the local client machine, with no more references to the http server.[2]

4. Long Term Aspects

At this point in time there are limitations to what can be done with this technique, mainly having to do with security considerations: it is impossible for an applet to access a local file. Another limitation is that Java is an interpreted language, and, as such, is 20 times slower than "C." However, these limitations apply not only to astronomy, but to other fields, as well. We can, therefore, expect that the market will solve this problem for us.

This approach opens up a wide range of possibilities. For instance, the traditional approach of data archiving has been to archive raw data, plus data calibrated soon after the observations were carried out. Increasingly, we see a trend towards on-the-fly calibration, which means that the data are re-calibrated during the retrieval process using the most appropriate calibration reference files and the most recent version of the calibration software. The fact that even modest computers today are capable of performing the calibration of large data sets in a reasonable time makes this approach feasible. It is quite conceivable that future observational facilities will soon eliminate the storing of calibrated data.

The on-the-fly calibration is being performed at the archive machine, and, so far, we have been able to support this service. With the increased appreciation of data archives, and with more sophisticated and larger data sets, it is quite evident that the archive machine will become hopelessly overloaded. Applets provide the elegant solution of pushing the computational load out to the client site.

Acknowledgments. Thanks are due to Richard Hook and Michael Naumann for solving a number of Java related installation problems.

References

Albrecht, R., et al. 1997, this volume, 443

Pirenne, B., Benvenuti, P., & Albrecht, R. 1997, this volume, 290

[2]http://archive.eso.org/wdb/wdb/preview/preview_hst/query/Z0E35308T