Java Analysis Studio

A.S. Johnson[1]

[1] SLAC, PO Box 4349, Stanford University, CA 94309, USA

Abstract

This paper describes version 2 of Java Analysis Studio, an experiment independent data analysis tool for High-Energy physics. We describe in particular the JASHist bean for displaying histograms, and improvements to the GUI included since the previous release of JAS. We will also discuss the experiences of some experiments that are currently using JAS including Babar and the Linear Collider Detector (LCD) group.

keywords JAS, Java, Analysis, Histogram, XML

1. Introduction

Java Analysis Studio (JAS) is a graphical application for analysis of high-energy physics data. The application is independent of any particular data format, so that it can be used to analyze data from any experiment. The application features a rich graphical user interface (GUI) aimed at making the program easy to learn and use, but which at the same time allows the user to perform arbitrarily complex data analysis tasks by writing analysis modules in Java using the built in editor/compiler. The application can be used either as a standalone application, or as a client for a remote Java Data Server. The client-server mechanism is targeted particularly at allowing remote users to access large data samples stored on a central data center in a natural and efficient way. JAS is written entirely in Java and will run on any platform with a Java virtual machine. The basic functionality of Java Analysis Studio 1.0 was covered at the previous CHEP conference, so given the limited space/time available this paper will cover primarily additions and improvements implemented since then and incorporated into release 2.0. Readers interested in more detail are referred to our CHEP 98[1] paper or to our web site at http://www-sldnt.slac.stanford.edu/jas

2. The JASHist Bean

One of the key components of Java Analysis Studio is the JASHist bean, which is responsible for the display of histograms and scatter plots. Since the previous release of JAS support for 2-D histograms has been added by integrating the 2-D plot features of the Babar JavaHist package[2].

The JASHist bean is designed using the model-view-controller pattern, so that data to be displayed need only implement a simple java interface and need have no other dependence on the JAS package. This makes interfacing arbitrary data to the plot bean very straightforward. Care has been taken in the design and implementation of the JASHist bean to ensure that it is a modular component that can be used easily in other applications. A chapter in the JAS Users Guide describes how to do this[3].

The current JASHist bean includes support for:

- Display of 1-D histograms, 2-D histograms and scatter plots. Scatter plot support is optimized to handle up to millions of points.
- Overlaying of several histograms or scatter plots on one plot.
- Interactive fitting of arbitrary functions to 1-D histograms.

- Numeric or time axes, plus axes with named bins.
- Many display styles that can be set interactively or programmatically.
- Dynamic creation and display of slices and projections of 2-D data.
- Direct user interaction, by clicking and dragging.
- Data that is constantly changing, including very efficient redrawing to support rapidly changing data (handles over 100 updates/second).
- Printing using both Java 1 and Java 2 printing models. High quality print output is available when using Java 2.
- Saving plots as GIF images or as XML. Support for encapsulated postscript and PDF is in progress.
- Custom overlays which allow data to be displayed using user defined plot routines for specialized plots.

2.1. Servlet Support

One way in which it is possible to use the JASHist bean is in a Java servlet[4]. Java servlets provide functionality vaguely similar to Java applets, except that whilst an applet runs in the browser (client) the servlet runs on the web server, and sends its contents to the browser either as HTML or as an image. While an applet has some advantages, especially if the data to be displayed needs to be updated rapidly, servlets are generally much easier to setup since they do not rely on the browser (correctly) supporting Java. The model-view architecture used by the JASHist bean means that typically all that is needed to implement a particular servlet is a simple adaptor to convert the actual data source to the JASHist DataSource interface. Instructions illustrated with several examples are available in the JAS servlet How To[5].

2.2. XML Support

The JASHist bean is able to read and write plots as Extensible Markup Language[6] (XML) files. XML is a markup language similar to HTML except that it allows arbitrary tags that can be defined in order to efficiently and accurately represent data in any particular problem domain. The XML format supported by the JASHist bean has been designed to be independent of the underlying JAS architecture, in the hopes that it can form the basis of an HEP wide mechanism for the exchange of histograms between applications. Since XML is an ASCII format it is also possible to write histogram descriptions by hand or to generate them from a script. More details in the XML support in JASHist are available in the JAS XML How To[7].

2.3. 3-D Scatter Plots and Lego Plots

Andrey Kubarovsky and Joy Kyriakopulos at Fermilab have been using some of the experience they gained in working with HistoScope[8] to develop Lego plots and 3D surface and scatter plots in Java using Java's 3D API[9]. Prototype implementations of these 3D widgets exist, and we are currently working with them to integrate the 3D functionality into the JASHist widget. Since the Java 3D API is a Java "standard extension" rather than a part of the core Java library, we will exploit Java's dynamic code loading capabilities to make sure the functionality gracefully degrades in the case where the 3D API is not available.

3. The Java Analysis Studio GUI

Java Analysis Studio features a full-featured Graphical User Interface including a built in editor for writing Java Analysis routines and built in compiler/loader. The graphical user interface also features a complete help system, wizards to help new users get started, facilities for viewing and manipulating plots, and is extensible via "Plugins" written in Java to provide user or experiment specific features.

3.1. JEdit Syntax Highlighter/Editor

In the previous release of JAS we used an extension of the Swing Editor Pane to provide a simple Java editor with syntax highlighting. We have now switched to using the SyntaxTextArea bean written as part of the JEdit editor[10]. By incorporating this bean we have been able to get markedly improved functionality with little effort on our part (thanks to the hard work of the JEdit authors). In addition we plan to integrate more of the JEdit features in the future, such as text search and replace, and also hope to be able to support JEdit plugins. Since many JEdit plugins are being developed, including plugins that support automatic code completion and IDE features such as integrated debugging, this should enable us to rapidly add very useful additional functionality to the JAS framework.

3.2. HTML Page Display

Another powerful feature that we were able to add with relatively little work by exploiting existing Java functionality is the ability to display HTML pages within JAS. The JEditorPane within Java's Swing toolkit is able to display most HTML 3.2 pages, and can be extended to support custom embedded Java objects. We have used this functionality to make it possible to embed JASHist plots within HTML pages, using the HTML OBJECT tag. This functionality allows us to embed tutorial information and demos within JAS, and can also be used in online monitoring applications to display predefined pages of plots to users.

4. Online monitoring API

JAS supports a number of API's designed to allow extensions to be build without having to delve into the internals of the JAS application, and in such a way that they are likely to continue to work with future releases of JAS. We have recently added a new API specifically to support online monitoring applications. The API provides for:

- Interfacing the JAS server to a pre-existing set of histograms which are updating in real time.
- Communication between the client and the server so that Plugins installed in the server can be used to send commands to the server.

This API is described in more detail in the JAS Online Monitoring How To[11].

5. Open Source Model

JAS is now an open source project, with source code browsable directly from the JAS web site (using jCVS servlet[12]), or accessible using any CVS client. The instructions for gaining read-

only access to the CVS repository are available on the JAS web site, and read-write access is available to registered developers. Our intention is to continue to refine the design of JAS to make it easier to integrate with other applications and our hope is that making the source available will make it easier for other to understand how it works, and to contribute fixes and improvements.

In order to further facilitate cross-platform development we have adopted jmk[13], a pure Java utility similar to make. This enables JAS to be built on any platform with a Java Development Kit (JDK) available.

6. Examples of Use

6.1. Linear Collider Detector

The US Linear Collider Detector (LCD) group has build an entire reconstruction and analysis framework in Java, which can either be run standalone or inside Java Analysis Studio. As part of this effort some standard 3-vector, 4-vector, event shape and jet finding routines were developed and these have subsequently been integrated into the physics utility section of JAS 2.0.

The LCD group has used the "Plugin" functionality of JAS to provide an event display that can be run inside JAS and automatically adapts to different detector geometries. The LCD group has also set up a central data repository at the University of Pennsylvania running the Java Data Server software provided with JAS so that physicists anywhere can use the JAS client to connect to the Penn server and analyze the data stored there.

6.2. Babar

Babar is using JAS as a means of presenting online monitoring histograms to physicists on shift in the Babar control room. They use a three-tier approach using a server that acts as a gateway between their CORBA based distributed histogram facility and JAS's RMI based client/server communication protocol. The server is implemented in Java and uses the JAS online monitoring API. Histograms are displayed in the JAS client using a HTML pages with embedded "live" plots for each detector subsystem. The HTML pages also provide descriptions of the plots, and contain hyperlinks to additional pages with more detailed diagnostic histograms. Babar also uses the custom overlay feature of JASHist to provide specialized plots such as online "scalers". Code for the custom overlays can be dynamically downloaded from the server to the JAS client so there is no need for special software to be installed on the client.

7. Java Performance and Experience

Our experience in using Java is that it is an extremely good language for developing GUI applications and for rapid prototyping of analysis and reconstruction tasks. We have found the cross-platform compatibility to be excellent, with most development being done under NT, and the resulting code "just working" under Unix (several people have also reported success at running JAS on the Macintosh).

Java performance has continued to improve over time. IBM has recently released versions of Java incorporating their optimized just-in-time compiler that run under AIX, Linux, OS/2 and

Windows and which give almost a factor of 10 improvement in speed over earlier Java implementations. Experience with the LCD reconstruction code has shown that even pattern recognition and fitting code can be implemented in Java and give very competitive performance.

The rapid development of Java has not been without problems however, as the rush to add more functionality to the language has sometimes left more mundane features and bug fixes languishing. One area where Java has so far been weak is in the printing arena, where until recently it was very hard to produce high quality print output. The printing engine is dramatically improved in release 1.3 of Java, and once this becomes widely available this particular limitation should be fixed.

One cloud on the horizon is Sun's failure to act on their earlier commitment to submit Java for international standardization. Unfortunately this may limit people's willingness to rely on Java for projects with the extremely long lifespan of today's large HEP experiments.

Acknowledgements

Most of the features described in this paper have been implemented by Peter Armstrong, Kevin Garwood, Jonas Gifford and Azhar Zuberi, students working at SLAC from the University of Victoria. LCD and Babar collaborators including Gary Bower, Kevin Rennert and Alex Samuel have also made significant contributions.

References

[1] Java Analysis Studio, paper published in the CHEP 98 proceedings, http://www-sldnt.slac.stanford.edu/jas/documentation/Chep98/Chep98.htm
[2] Scott Metzler and Alex Samuel, http://www-sldnt.slac.stanford.edu/hepvis/Papers/Web/19/dhpcorba.html
[3] Using the Plot Widget in Your Own Applications, http://www-sldnt.slac.stanford.edu/jas/documentation/usersguide/jashist/default.shtml
[4] See the Java Servlet specification - http://www.javasoft.com/products/servlet/2.2/
[5] JAS Servlet How To, http://www-sldnt.slac.stanford.edu/jas/Documentation/howto/servlet/default.shtml
[6] See the XML specification at: http://www.w3.org/XML/
[7] JAS XML How To, http://www-sldnt.slac.stanford.edu/jas/Documentation/howto/xml/default.shtml
[8] Histo-Scope Plotting Widget Set, http://www.fnal.gov/fermitools/abstracts/plotwidgets/abstract.html
[9] Java 3D™ API Specification, http://www.javasoft.com/products/java-media/3D/forDevelopers/j3dguide/j3dTOC.doc.html
[10] JEdit is a pure Java editor, by Slava Pestov, http://www.gjt.org/~sp/jedit.html
[11] JAS XML How To, http://www-sldnt.slac.stanford.edu/jas/Documentation/howto/online/default.shtml
[12] jCVS is an open-source CVS client written in pure Java. JCVS servlet provides a way of making CVS repositories browsable via the web. http://www.jcvs.org
[13] jmk - Make in Java, http://www.ccs.neu.edu/home/ramsdell/make/edu/neu/ccs/jmk/jmk.html