ADASS XIII
ASP Conference Series, Vol. XXX, 2004
F. Ochsenbein, M. Allen and D. Egret eds.

VO Enabled Mirage and the IVOA Client Package

Samuel Carliles[1], Tin Kam Ho[2], William O'Mullane[1]

Abstract. Astronomers commonly analyze astronomical data by imposing different views on the data, frequently viewing it in image form or as multi-dimensional plots. The ability to correlate the data in these views in order to see manifestations of patterns across views would be a powerful tool. The Mirage data visualization application offers this functionality. In order to increase the value of this tool to astronomers, we have added two features to Mirage, namely a module for viewing FITS images, and the ability to load VOTable data. During the process of adding the VOTable functionality to Mirage, we also developed a separate Java package, called the IVOA Client Package, which can be integrated easily into any other Java application to provide the ability to load VOTable data via Cone search queries submitted to any Cone services published in a Registry, or by a direct SDSS CAS query. We describe briefly the IVOA Client Package, the usage of VO Enabled Mirage, and the process of writing a data view module which can be incorporated into Mirage. We also briefly describe the process of programming a FITS viewer in Java using JSky.

1. Introduction

We describe the IVOA Client Package, a Java package which offers high-level VO functionality to any Java application. We then describe VO Enabled Mirage, an extended version of Mirage[3] (Ho 2003), which offers easy retrieval and analysis of VOTable data, and correlation with FITS image data. We also describe the process of integrating a new data view module with Mirage, using the example of the `TwoDPictorialDisplayMode` module, which allows viewing of FITS images, and which includes some common astronomical image manipulation functionality, using components from JSky[4] for some operations.

[1]Johns Hopkins University, Baltimore, Maryland

[2]Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey

[3]`http://cm.bell-labs.com/who/tkh/mirage/index.html`

[4]`http://archive.eso.org/JSky/`

2. The IVOA Client Package

The IVOA Client Package[5], used by VO Enabled Mirage, can be integrated easily into any Java application, enabling retrieval of VO data. VOTable[6] parsing is based on SAVOT[7] or optionally on JAVOT[8]. The IVOA Client Package is composed of three main components: a search panel which provides a user interface to VO Cone/SIAP and SDSS CAS searches, a common interface, called `ivoa.VOTWrap`, for accessing VOTable data from SAVOT and JAVOT interchangeably without requiring different application code, and a Task Manager which gives user control over indeterminate-length operations.

The `VOTWrap` implementation supports access to table data in VOTables. Support for other features of VOTables may be implemented as needed. The motivation behind interchanging SAVOT and JAVOT is that they behave slightly differently, and the optimal choice depends on the situation. SAVOT is a very tolerant parser which will frequently parse non-compliant VOTables, but it doesn't propagate exceptions up to application code, so graceful exception-handling with SAVOT is not possible. JAVOT does propagate exceptions to the calling application, but it's very strict about standards compliance. In particular, we discovered that JAVOT will not parse VOTable data if the VOTable refers to its Schema definition and for some reason the definition is inaccessible (for instance, due to server outage.) There does not appear to be any way to relax this behavior.

The Task Manager has been designed to handle any kind of task that can be programmed, with the currently implemented tasks tending to be of the data-retrieval variety. The `ivoa.Task` class can be extended easily to provide whatever functionality is necessary.

3. VO Enabled Mirage

VO Enabled Mirage[9] has all the features of "classic" Mirage including multiple data views and clustering algorithms, with the additional feature of being able to load VOTable data and perform VO Cone/SIAP and SDSS CAS searches. It also includes an astronomical imaging module which loads FITS images using JSky classes, and allows image manipulation operations common in many astronomical imaging applications.

3.1. Using VO Enabled Mirage

Data files, including VOTable files, may be loaded into VO Enabled Mirage from the `Console` → `New Dataset` menu item just as in classic Mirage.

---

[5] `http://skyservice.pha.jhu.edu/develop/vo/ivoa/`

[6] `http://www.us-vo.org/VOTable/index.html`

[7] `http://simbad.u-strasbg.fr/public/cdsjava.gml`

[8] `http://www.us-vo.org/VOTable/JAVOT/`

[9] `http://skyservice.pha.jhu.edu/develop/vo/mirage/`

VOTables may also be loaded directly from Cone and SDSS CAS searches using the `Console` → `New Dataset from VO Source` menu item. Once a dataset is loaded, the astronomical imaging module can be opened by dragging the M51 icon () into any data view panel as in classic Mirage.

3.2. Adding VOTable Functionality

Mirage is implemented as a command interpreter with a programmatic interface which can be called by user interface code. The user interface to classic Mirage instantiates a different Mirage interpreter for each dataset. VO functionality was added to Mirage by extending the original user interface. Data loading commands generated by the user interface are intercepted and converted from VOTable to Mirage's native data format if necessary. Then adding a menu option to invoke the IVOA Client search panel was possible using the normal Java Swing API. The code for converting VOTable data to Mirage native data was written using the `VOTWrap` SAVOT/JAVOT wrapper interface from the IVOA Client Package which allows generic access to field information, some of which must go at the beginning of Mirage's file format. Then data is written out one record per line with space-delimited fields.

3.3. Adding a Data View Module

The `ActivePanel` Interface

The `mirage.MirageGraphics.ActivePanel` interface can be implemented by any module that one wishes to integrate with Mirage. This interface is still under development, and the semantic intepretation of these methods is still not clearly defined. The interpretation used to implement the `TwoDPictorialDisplayMode` module is offered here. Implementing classes must provide concrete implementations of the methods listed below. The `mirage.MirageData.Entry` class encapsulates a row of data, and is used in many of these methods. Each `Entry` also has a `Color` associated with it, which may be assigned by the user.

`public Vector getSelected()`: This should return a `Vector` containing instances of `Entry` representing the rows currently selected.
`public void clearSelected()`: This tells the module to deselect any currently selected `Entry`s.
`public void transferDropTarget(DropTarget dropTarget)`: This should invoke `setDropTarget(dropTarget)` on any `Component`s in the module so that the module can be overwritten by another module selected by the user.
`public void colorDataEntry(Vector entry)`: This tells the module to color the display of `Entry`s in `entry` as appropriate for the view.
`public void highlightDataEntry(Vector entry)`: This tells the module to highlight the `Entry`s in `entry` as appropriate for the view.
`public void clearColors()`: This tells the module to show all `Entry`s in the default `Color` for multi-color display.
`public void clearHighlights()`: This tells the module to show all `Entry`s in the default `Color` for monochrome display.
`public void changeToMonochrome()`: This tells the module to show selected `Entry`s in the monochrome highlighted `Color`.

`public void changeToColor()`: This tells the module to show selected `Entry`s using their associated `Color`s.
`public void alternateVariables(int mode, int row, int col, int idxCell, int nCells)`: This method informs the module of its position within a matrix of similar modules. The module can alter its appearance accordingly (e.g. select a different variable for display). For image data the display dimensions are fixed, so in `TwoDPictorialDisplayMode` it is implemented as a no-op.

To load a new module into Mirage, add a line reading

`externalpanel` *`classname iconimage`*

to the `Property.dat` file that comes with Mirage, where *`classname`* is the fully qualified name of the class for the module to be loaded, and *`iconimage`* is a path sufficiently specific that the icon image file can be loaded as a resource at runtime by the Java Virtual Machine.

The `TwoDPictorialDisplayMode` Data View Module

The `TwoDPictorialDisplayMode` in VO Enabled Mirage allows the user to load FITS images, apply user-selected cut levels, apply colormaps, and adjust brightness and contrast with SAO DS9-style controls. Also, being a Mirage `ActivePanel` class, it responds to Mirage broadcast messages, and can itself broadcast messages to other Mirage modules. Much of the implementation uses classes and code from JSky, which provides an easy way to load FITS images and use them within the Java Advanced Imaging API (JAI) framework, as well as many image processing operations and support for WCS to image (and reverse) coordinate transforms. Some problems were encountered using JSky. Mapping from an arbitrary FITS image data type to a type associated with Java `RenderedImage`s isn't entirely predictable. The implementation of certain image processing functions had to be copied verbatim due to a very tight coupling between the model classes and the view/controller classes.

The `TwoDPictorialDisplayMode` image processing classes are implemented as a class hierarchy with each extension adding processing functionality to its superclass. The intention was to encapsulate the image processing functionality in the image itself to make it easy to integrate into any Java graphical component. However, this design resulted in slow performance as well as difficult maintenance and extensibility. We are currently redesigning it with the goal of improving each of these aspects.

Acknowledgments. We would like to thank Allan Brighton for his advice on using JSky in the implementation of the `TwoDPictorialDisplayMode` module.

References

Ho, T.K. 2003, in APS Conf. Ser., Vol. 295, Astronomical Data Analysis Software and Systems XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), [O6-2]