MSTAPP, A RICH CLIENT CONTROL APPLICATIONS FRAMEWORK AT DESY

Kirsten Hinsch, Winfried Schütte, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

The control systems for PETRA 3 [1] and its pre accelerators extensively use rich clients for the control room and the servers. Most of them are written with the help of a rich client Java framework: MstApp. They total to 106 different console and 158 individual server applications. MstApp takes care of many common control system application aspects beyond communication. MstApp provides a common look and feel: core menu items, a colour scheme for standard states of hardware components and predefined standardized screen sizes/locations. It interfaces our console application manager (CAM) [2] and displays on demand our communication link diagnostics tools [3], [4]. MstApp supplies an accelerator context for each application; it handles printing, logging, resizing and unexpected application crashes. Due to our standardized deploy process [5] MstApp applications know their individual developers and can even send them – on button press of the users - emails. Further a concept of different operation modes is implemented: view only, operating and expert use [6]. Administration of the corresponding rights is done via web access of a database server. Initialization files on a web server are instantiated as JAVA objects with the help of the Java SE XMLDecoder [7]. Data tables are read with the same mechanism. New MstApp applications can easily be created with in house wizards like the NewProjectWizard [5] or the DeviceServerWizard [8]. MstApp improves the operator experience, application developer productivity and delivered software quality.

INTRODUCTION

As part of the transition of PETRA 2 - an injector for HERA - to PETRA 3 - a dedicated synchrotron machine - an entire new control system was implemented [1], [9]. It is based on Java as a platform independent language, TINE [10] as a communication protocol and MstApp as an application framework.

The control system tasks are implemented with networked rich client JAVA SE 6 applications. We distinguish between console and server applications. And for some complex cases we also have middle layer applications.

Many aspects of these applications are generic. A framework called MstApp was designed to cover the generic aspects not related to the communication protocol. The application developers could then concentrate on the application specific aspects.

LOOK AND FEEL

A common look and feel of all accelerator control client applications eases the task of the operating crew. This is both true on the small level of standard colours for different component states, and on the large level like the layout of the entire application. Printing, protocols, help etc. are always found at the same place.

Standard Colours

There are always a lot of discussions about the best colours for different component states. And once there is an agreement how to find the documentation. The framework MstApp just integrates our agreement on colours as mnemonically named subclasses of the standard Java "Color" class. As an example the colour for error states is defined as a red background with a white foreground (a black foreground would be too difficult to read on a red background).

Standard Sizes and Positions

Applications are not alone on the display but have to live together with many other ones. Well defined application sizes help to use the screen better. MstApp gives predefined sizes and positions. Both can also be saved on a local basis to the hard disc or managed by our console application manager (CAM) [2]. This feature is useful for the control room by having standard work environments. It is also useful for server PCs with many servers. A missing small server display is spotted at a glance.

Standard Menu

Some standard functionality is already provided in the menus. The predefined layout is the following:

- File, containing show logging, printing and exiting
- Machine for the applicable accelerator (optional).
- Options, containing an extensive dialog
- Help, containing an about box, and paths to the most relevant pages of the accelerator control room WIKI

The application developer can add new menus, or in existing menus new menu items. The options dialog is also extensible.

Resizing

A high fraction of our application developers like to design with absolute layout. No layout manager is used. Since the size of our applications is related to the screen size, this works well as long as the size of our display monitors does not change. In reaction to the unavoidable change of display size, a so called ApplicationResizeManager was developed.

An activated ApplicationResizeManager resizes all components of the application top-down. The following properties are resized: position, width, height, font, tables and icons. AWT graphics has to be scaled manually in the paint method. Resizing the font the ApplicationResizeMa-

nager considers both, width and height of the component to calculate the new font size.

Auxiliary Information

A standard MstApp application contains at the bottom a status bar with information about the PC name, date, time, operations and server mode. This is especially helpful for graphical application copies to the logbook [11].

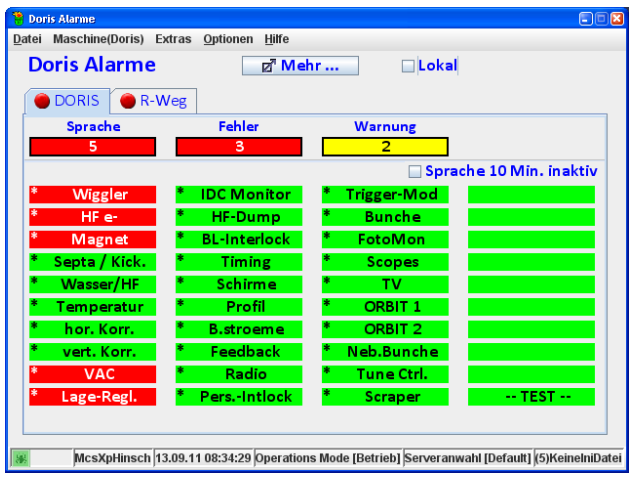

Figure 1: Typical MstApp based client application. The accelerator Doris is chosen. An application developer specific menu “Extras” is added. The status bar at the bottom shows the computer name, date and operations mode. At its left one finds the communication link diagnostic tool Spider.

COMMON TASKS

Operation Modes

Console programs for accelerator control have different users.

Our most important users are the operators in the control room. The operator needs good reading and writing access to the servers running the accelerator hardware.

Then there are users outside the control room interested in current accelerator parameters. They should not be allowed to change the accelerator hardware parameters. Their program use should be restricted to only viewing them.

Finally there are the hardware developer themselves, who need to access their specific devices in an expert way reading and writing to the corresponding servers far in excess of the operators.

Each of these views operating, viewing and expert mode is natively implemented in the framework. Maximum rights are enforced on a basis of user name, pc name and application name via a database table.

We also differentiate between console application accesses to the real hardware server and a simulation server.

These security measures are complementary to the ones inherent to the communication protocol. See for example [12].

Accelerator Context

There are applications so specific that they run only for a single accelerator. Others like the display of the dc current monitor are the same for all circular accelerators. We supply a context for all those accelerators and a standard switch mechanism between them.

Printing to Logbook and Developer E-Mails

The MstApp framework provides an identical print dialog frame for each application. This frame enables the users to print a screenshot to a normal printer as well as to create full logbook entries [11]. The user has a mask for his name, a category and some additional text. A screenshot of the application is added automatically. The user can print this entry to the accelerators logbook (default) and/or sent it as an E-Mail to the developer. To declare his choice he uses two checkboxes.

Help

Help is a very difficult topic. There has to be help at all, it has to be up to date and correct, and the user has to be able to find it. Some users might even want to improve on it.

Due to our common deploy process [5] a minimal up to date help within the application is possible. In this about box style help the user finds the application developers name, telephone number and email address. He finds the version, creation date and location of the application and all its referenced libraries.

If more help seems to be useful, the developer can register a page of our control room wiki. This will be offered by the framework to the user. Still someone has to create and maintain this page.

Console Application Manager (CAM)

For routine accelerator operation it is helpful to have a defined set of applications with defined places for the consoles. Our console application manager [2] does this for us. All MstApp applications are native CAM clients. No developer has to invest here any work.

Communication Link Diagnostics Tools

Network communications is never absolutely reliable. It is therefore mandatory to have some handy analysis tools for an application. The Spider [4] shows all direct TINE [10] links and Tarantula [3] shows the direct links, the links of the linked servers and so on up to a specified depth.

Unhandled Crashes

Obviously applications should never end abnormally (crash). In practice they do crash on some rare occasions. In this case it is important to end the application gracefully and to help the developer to improve on the code.

Our framework tries to ease on this. A Java shutdown hook catches those uncatched errors and tries to display a crash screen. The crash screen allows for printing a screen shot, the about box information and a comment to the accelerator control logbook. Also an email can be sent to the known application developer simple by the press of a button.

Applications do not only die by crashing. Sometimes they just freeze. This is especially bad for server programs. Here a watchdog [2] might stop and restart the server application. A screenshot is put into the TINE protocol server. All those screenshots can be viewed from any framework application.

Initialisation Files

The framework supplies a platform independent way of accessing files from a webserver. Initialisation files, some data tables and the data for allowed operation modes are transported in an xml encoded fashion that can be directly converted to a tree of Java objects with the help of the Java SE XMLDecoder [7].

Technically this works very well though the actual use is a little cumbersome.

Protocols/Logging

Writing protocols for accelerator applications has different objectives than the standard Java logging tools. Therefore a custom logging mechanism for application protocols was introduced. The last 500 (can be configured) entries are kept in main memory and the last seven days are kept on local disc for display directly from the application. Logging from the framework and the application and caught logging from the Java logging mechanism is tagged with a category. Logging of Java exceptions is also supported. By supplying a tag the logging message will be stored in the central TINE logger server [13].

STANDARD SERVER APPLICATIONS

The server framework provides a special JFrame for server applications. It ensures an identical look and feel. So the developers can concentrate on the work without graphical aspects. The server frame inherits from an MstApp client frame and provides all aspects the client has.

A server application has three resize modes. Normally it is small, just a square with the name, the icon and a state like idle or reading data. The small size allows many applications neatly aligned in rows.

Pressing the maximize button the application resizes to medium size. Here additional output shows the start time, the last commands and the last events. The entries to two lists for the commands and the events are done with the MstApp logging commands by using special category tags.

In expert mode the application resizes to the large size. It shows a pane, to which the developer is free to add test buttons and output features, which are interesting for the particular server. Other parts of the frame should not be changed.

The non-graphical work of the server application is coded in classes created by the DeviceServerWizard [8]. The developer adds device specific code in his own custom classes.

Figure 2: A server in small size. It only shows the name, the menu and an icon. The status bar is hidden.

INTEGRATION WITH OTHER FRAMEWORKS AND WIZARDS

Our framework interacts by design well with our other tools: The NewProjectWizard [5], our standard deployment [5] and the DeviceServerWizard [8]. It works with the communication protocols like TINE [10] and DOOCS [14]. Even the combination with jddd [15] is easily possible.

TECHNICAL NOTES

How is it realized in Java? The developer has to supply a special Java class that inherits from a special JFrame: MstFrameMain or in case of a server MstServerFrameMain. MstFrameMain supplies context information and the basic look and feel. Components from the developers are added only to the central user area. The framework creates the MstFrameMain and knows it via a "String" starting parameter. Also parameters for the application name, the accelerator name und application specific parameters can be supplied.

The logging is supplied with a static mechanism.

Standard features like change of operations mode, change of the accelerator and many more are supported by extensive use of the observer pattern [16].

The framework requires only the standard Java libraries. This eases a consistent deployment, since no library can be forgotten by the developer, leading to strange runtime errors. Elements from other libraries - like the Spider - will only be shown, if the hosting library is supplied by the developer (creation by reflection).

CONCLUSION

MstApp improves the operator experience, application developer productivity and delivered software quality. A

common look and feel is achieved and at the same time application developers are relieved from many mundane tasks. Common tasks are only written and tested once. Many changes to the environment can be adapted to in one place only.

ACKNOWLEDGEMENTS

The framework MstApp - as a simple central hook into the control system - has many more authors than this paper. There are major contributions from Reinhard Bacher, Piotr Bartkiewicz, Andreas Labudda, Marcus Walla and Franziska Wedtstein. It is a pleasure for us to thank all mentioned and unmentioned contributors.

REFERENCES

[1] Reinhard Bacher, "Commissioning of the New Control System for the PETRA 3 Accelerator Complex at Desy", Proceedings of ICALEPCS 2009, Kobe, Japan.
[2] Piotr Bartkiewicz, DESY, Hamburg, private communication
[3] Marcus Walla, DESY, Hamburg, private communication
[4] http://pub.cosylab.com/acop/site/AcopSpider.html
[5] Andreas Labudda, "Building and Deploying loosely coupled Console Applications", Proceedings of PCaPAC 2006, Newport News, USA, p 126
[6] Jürgen Maaß, Georg Mann, "PETRA III Operation Modes", DESY MCS1 internal paper 2005-02-04
[7] http://java.sun.com/products/jfc/tsc/articles/persistence4/
[8] Josef Wilgen, DESY, Hamburg, private communication
[9] Rüdiger Schmitz, "What's Behind an Accelerator-Control System?", Proceedings of PCaPAC 2010, Saskatoon, SK., Canada.
[10] Philip Duval, http://tine.desy.de/
[11] R. Kammering et al., "E-logbook Reloaded - or the Renovation of DESYs Electronic Logbook", Proceedings of ICALEPCS 2009, Kobe, Japan.
[12] Philip Duval, http://adweb.desy.de/mcs/tine/TineSecureServices.html
[13] Philip Duval, TINE Release 4.0 News, April 18, 2008; http://adweb.desy.de/mcs/TINE_Users_Meeting/ 2002Apr18/Release4News.pdf
[14] http://doocs.desy.de/
[15] http://jddd.desy.de/
[16] E. Gamma et al., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, 1994