HEPBOOK - A PERSONAL COLLABORATIVE HEP NOTEBOOK

G. Roediger, Corporate Computer Services Inc., FNAL, Batavia, IL 60510, USA
P. Pomatto, Corporate Computer Services Inc., FNAL, Batavia, IL 60510, USA

Abstract

A High Energy Physics experiment has between 200 and 1000 collaborating physicists from nations spanning the entire globe. Each collaborator brings a unique combination of interests, and each has to search through the same huge heap of messages, research results, and other communication to find what is useful.

Too much scientific information is as useless as too little. It is time consuming, tedious, and difficult to sift and search for the pertinent bits. Often, the exact words to search for are unknown, or the information is badly organized, and the pertinent bits are not found. The search is abandoned, the time is lost, and valuable information is never communicated as it was intended.

Much of collaboration's information is in the individual physicist's paper logbooks. The physicists record important and pertinent information for their research. They save the log books to refer to it later, copy pages, and distribute them to their collaborators who share their interest and research.

Electronic Logbooks are now used in the control room of large detectors during the data acquisition phase. They have proven useful for communicating the status of the detector and to keep the history of lab sessions in a format that can be queried and retrieved quickly. It has enabled remote monitoring of the detector and remote emergency help.

We have implemented an electronic Control Room Logbook, called CRL. It is used in the D0 experiment's detector control room for the Run II acquisition. As of mid April 2004 there are over 305,000 entries in the D0 logbook, all viewable and able to be annotated from the web. Other experiments such as CMS, MiniBoone, and Minos have also adapted the CRL. These experiments all have very different needs, so each group configured and customized the CRL in many different ways.

The HEPBook will move the logbook from the control room to the personal and collaboratory HEP notebook. In this paper we will review the HEPBook technology and capabilities and discuss the new HEPBook architecture. Among the topics discussed will be the use of Java reflection to recursively produce an XML representation of an entry, the ability to save personal entries as well as share entries among a collaboration through multiple repositories which incorporate software agent technology, interface with the GRID, and implement multiple security models. The HEPBook runs on all Java platforms including Apple, Win32, and Linux.

KNOWLEDGEBOOK PROJECT

The project that created the HEPBook was funded by a phase II SBIR from the Department of Energy. The goal was to create a collaborative notebook useable in a scientific and commercial setting. It was to allow collaborators a means to store information and later share and distribute that information in an organized manner. The electronic notebook created is known as KnowledgeBook and the HEPBook is a KnowledgeBook configured with High Energy Physics customization. The HEP features are a result of comments from the HEP community that used the Control Room Logbook and physicists that provided feedback from using beta copies of the KnowledgeBook (KBook).

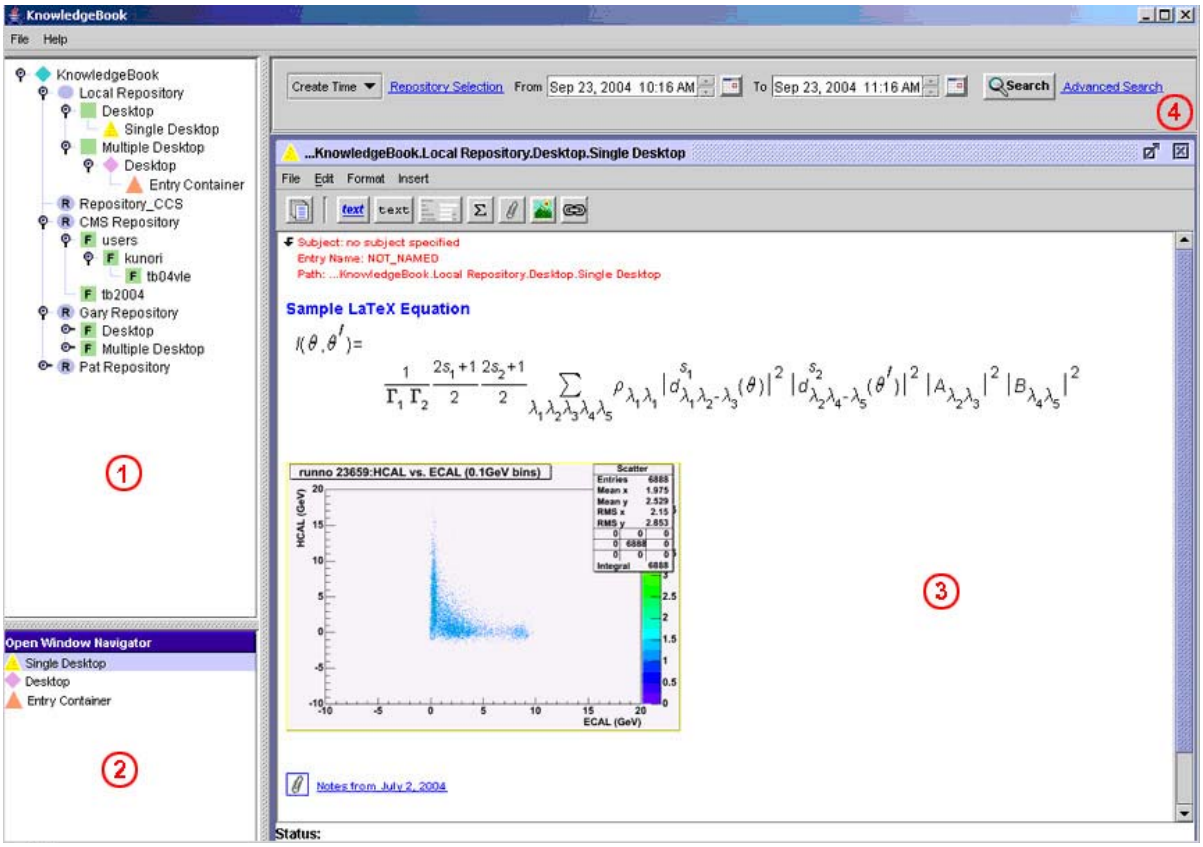
HEPBOOK FROM A USER VIEWPOINT

A collaborator can use the HEPBook as a private personal notebook and as a shared notebook in a public environment. The notes that are private are kept in a local repository configured on a filesystem accessible from the machine running the HEPBook. The remote repositories are accessed via a network configuration. The shared use of the notebook can be to one or more remote repositories associated with a group of users. This group may be as large as a collaboration or as small as 2 or 3 colleagues.

The client side software can be configured to run on Win 32, Linux, or Mac environments. There are dual boot users that share filesystems across multiple OS platforms. Since the local repository is stored in an OS neutral form, a dual boot user can view the entries from either system. Another popular configuration is to install all the files including the HEPBook in an AFS filesystem. Users then do not have to install the HEPBook and can reference their local repository from any machine with AFS.

The GUI

A user will see four regions in the HEPBook GUI (see Figure 1). The upper left region (#1), is a tree representation of the repository(s) structure. The node for the local repository appears just below the tree root, *KnowledgeBook*. Any attached remote repositories will appear at this same level. These available repositories can be exploded and explored to leaf nodes called containers. The user can create any structure and naming convention under their repository area to organize their entries. The



entries are placed into the container nodes. Right clicking a container node gives the option to explore the container's entries. Opening a container node brings up a blank entry on the desktop region of the GUI (#3 of Figure 1). Multiple containers can be opened at one time and are stacked on the desktop. The navigator (#2 of Figure 1), can be used to focus stacked containers and bring them to the top of the desktop. The last region is an expandable search form located at #4 above the desktop.

In Figure 1 the desktop region (#3) shows an entry being created. There are numerous built in entry components that can be added to an entry. These include plain mono spaced text, styled text, output from programs or text files, user defined XML forms, latex equations, attached files, images, and links to other entries or URLs. This list of components can be expanded as experiments or users create new information needs. An example of a new component would be recording sound bites and inserting them into the entry. The styled text can be formatted with color, font type, font size, bolding, underlining, or italicising. Text manipulation has cut, copy, paste, undo and redo.

The user defined forms are written in XML. They allow for input of structured data through fields, tables, radio buttons, checkboxes, scripts, lists, and pull down lists. Forms can also contain embedded static and dynamic forms to create more complex forms. Variables can be associated with form fields which can later be referenced in scripts. A script could be a shell script to submit a job to the local operating system. Fields within a form could be filled in and then the script with the variable values substituted can be executed on the local system. Because the fields within the form are part of the entry, saving the entry saves your job parameters for later access, sharing or editing.

Once an entry has been created it is named by the user and saved in a container. If this is a previous entry that is being edited, the user can decide to either overwrite the original, version the entry, or save it under a new name. Entries may be moved or copied between containers.

Remote Repository

Remote repositories are managed as a server with accounts given to users for access. A user can connect to the remote repository by providing the necessary security credentials. Each remote repository can specify its own security requirements. This means a user connecting to 2 or more remote systems may have multiple security credentials. The security model is based around JAVA JAAS.

The HEPBook uses different Java classloaders to separate repositories. This allows two different

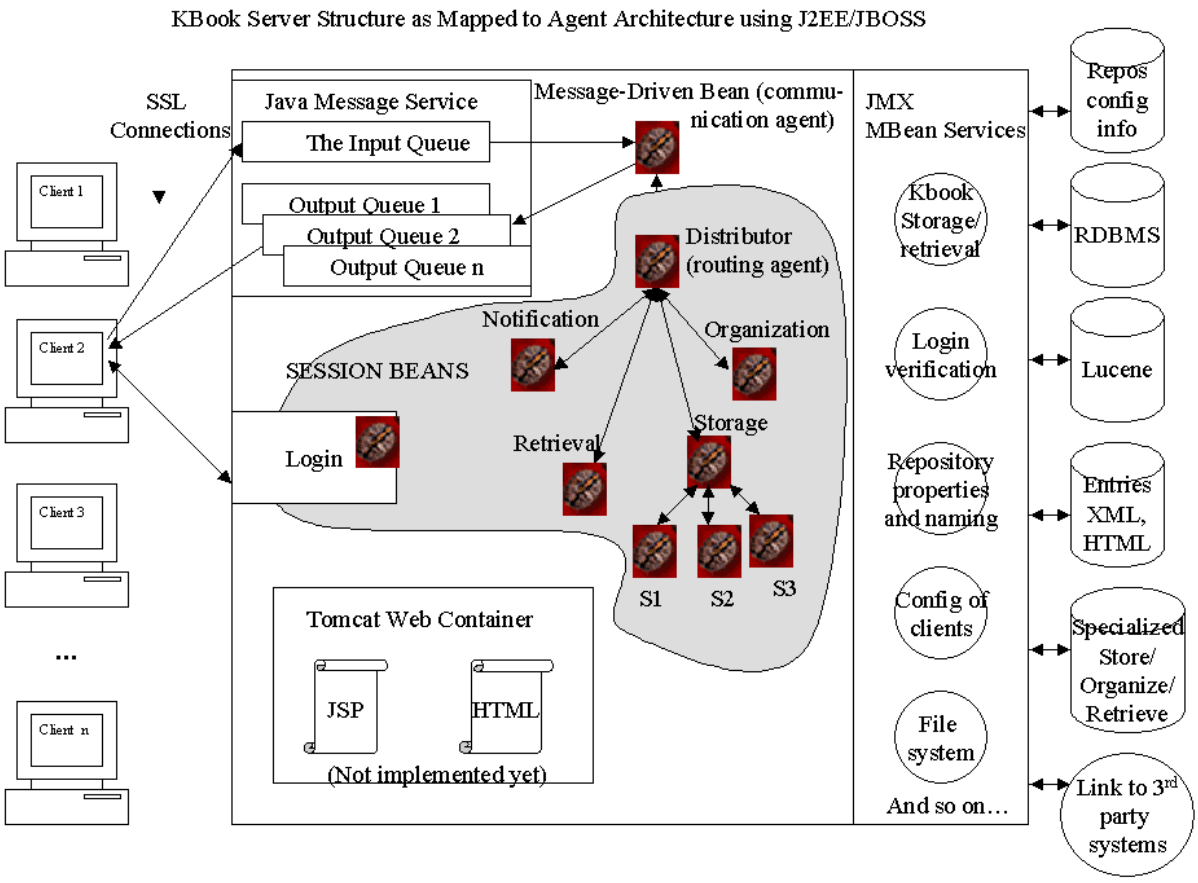


Figure 2 Architecture

repositories to run different versions of the code or have custom entry components available to entries for that repository only.

A user can connect to any remote repository where they have account authorization. A successful connection includes the synchronization of all structure changes and code updates that may have occurred while offline. All communication to remote sites is via SSL (Secure Socket Layer) channels.

Client Server Architecture

Figure 2 shows the client server architecture. The server is implemented using JBOSS, an open-sourced Java Enterprise Application Server (J2EE) container. The repository server allows collaborators in a virtual organization (VO) to share notes, files, code, data results, or any other information placed in an entry. Since entries can contain entry components that can be created as information needs change, this architecture allows for current as well as future information requirements. The user owns their entries that are placed in the central VO repository (VOR) and can place access restrictions on other collaborators.

The VOR will store entries the same as a user stores his personal local entries. Additionally the VOR will add access control so these entries may be selectively shared among the valid VOR collaborators. The VOR may be configured to handle entries in "special" ways depending on the data types or data values of entries' components. Both storage and retrieval of entries can involve custom handling of the request. This handling is determined by software agents programmed and administered by the VOR. In Figure 2 these agents are depicted by the S1, S2, or S3 agents connected to the Storage agent. The Storage agent will perform the same storage functions as the local repository but will also ask the specialized storage agents if they want to store additional content or meta data. This may involve the specialized agents contacting a 3rd party system such as a document database already in existence, or any other analysis database. The HEPBook provides a consistent interface and front end for existing systems while providing a clean mechanism for adding additional interfaces in the future. Retrieval and notification have similar agents and interfaces. Notification could involve email, posting to web sites, paging, or web services for example.

A TYPICAL SENARIO

The HEPBook has many capabilities and functions that solve the everyday tracking of what was done and how it can be recorded and shared. Following is a scenario based on using built-in HEPBook functionality to solve a CMS issue:

CMS has numerous run scripts that collaborators use to submit jobs to the CMS User Analysis Farm (UAF) at Fermilab. One such script is the CMKIN script that has various settings internal to the script that can be modified by the physicist running the job. This CMKIN script was inserted into a HEPBook form entry component as a XML form script. Additional fields, select boxes, and lists were added to the form to allow physicists to select values for various CMKIN parameters. These form values were then related to the embedded CMKIN script parameters through the XML. The user of the HEPBook could now create an entry in his local or CMS shared repository and add this CMKIN form as part of that entry. After filling in the form fields representing CMKIN parameters (zmass for example), the user could submit the job from the HEPBook directly to the UAF for processing. When the entry is saved in the HEPBook all parameter values and custom modifications made to the CMKIN script are also stored. At a later time the user could retrieve or share this entry with all the settings for the run. The entry could be modified with results from the CMKIN run by adding more information to the HEPBook entry. It would also be possible to change the entry fields and resubmit the job. Now the entry could be overwritten, versioned, or saved as an entirely new entry.

The CMS repository could be programmed with a software storage and retrieval agent to handle entries containing forms that contain CMKIN runs. The fields within the form could be saved and later analysed by specialized software storage, retrieval, and organizational agents within the VOR. These specialized software agents could also interface with custom CMS systems.

SUMMARY

The KnowledgeBook Project has succeeded in producing an electronic notebook capable of meeting the needs of many disciplines, specifically High Energy Physics. Through the use of plug-in data types (entry components) and specialized storage, retrieval, and organizational agents the notebook can be customized to meet the growing needs for storing and sharing collaborative information. Information that is saved in organized storage can be retrieved as knowledge.

ACKNOWLEDGEMENTS

The HEPBook was developed and funded through a phase II SBIR from the Department of Energy. The features and capabilities reflect the feedback and suggestions from users of the Control Room Logbook in various HEP experiments including D0, Minos, MiniBoone, CMS, and BTeV. Specific HEPBook input came from Thomas Phillips, Hans Wenzel, Shuichi Kunori, Patty McBride and many members of the Computing & Engineering for Physics Applications (CEPA) organization in the Fermilab Computing Division.

REFERENCES

[1] http://java.sun.com/

[2] http://www.uscms.org/scpages/general/users/farm/CMS_UAF.html

.