The Florida State University

College of Arts and Sciences

Java Servlets and User Interfaces for On-line Database

By
Wenzhong Cao

September 2001

A project submitted to the Department of Computer Science for the degree of Master of Science in Computer Science

Project Committee Members:

Dr. Greg Riccardi (major professor)

Dr. Lois W. Hawkes

Dr. Xiuwen Liu

Table of Contents

List of Figures 2
Abstract 3
1 Introduction 4
2 Search Process 10
2.1 Runs Search 10
2.2 Log Entries Search 11
3 Response Process 14
3.1 Run Summary 14
3.2 Log Entry Summary 15
4 Database Statement Execution 16
5 Search Runs On A Day 18
5.1 Run history List Page 18
5.2 Run Number Report For One Month 18
6 View Detail 19
6.1 A Run's Detail 20
6.2 A Log Entry's Detail 21
7 View Thread Headers 23
8 Thread Summary Page 25
9 Log Entry Update 26
9.1 Update Form Page 26
9.2 Update The Log Entry 27
9.3 Thread Update 28
9.3.1 From Unthreaded to Threaded 28
9.3.2 Stand Alone 29
9.4 Insert Thread 29
10 Log Entry Insert 30
11 Servlets Communication 32
12 Summary 33
Acknowledgement 34

List of Figures

Figure 1. Three-tiered Web site 4
Figure 2. Four servlets in the 3-tiered information application: (a) Run Information, (b) Log Entry Information, (c) Log Entry Update and Insert 6
Figure 2 (b). Log Entry Information Servlet 8
Figure 2 (c). Log Entry Insert and Update Servlets 9
Figure 3. Run summary search interface 10
Figure 4. One Runs search result under condition: b.start_date >= '05-Feb-2001' AND b.start_date < '06-Feb-2001' 11
Figure 5. Log Entry Information search interface 12
Figure 6. Log entry search result under condition: `L.entry_date >= '05-Jul-2001' AND L.entry_date < '06- Jul -2001' AND L.run >= 10338 AND L.run <= 10338` 13
Figure 7. A history list for Runs 18
Figure 8. A calendar for Runs in January 1999 19
Figure 9. The main steps for a Run's detail response process 20
Figure 10. The detail page for Run 27621 21
Figure 11. The detail page for log entry 3207 22
Figure 12. Threads and their headers 23
Figure 13. Threads headers' summary page 24
Figure 14. Thread 3207 summary page 25
Figure 15. Update form page for log entry 3207 27
Figure 16. Join to a thread. (a) Insert into an existing thread. (b) Create a new thread together with an unthread log entry. 28
Figure 17. A new form for a log entry insert 30

Abstract

My project is an information system. It is a 3-tiered application: client tier, servlet tier, and database tier. The application is developed in the Java language using Java servlets and JDBC.

The application addresses the need of the users of the Hall B experimental facility at Jefferson Lab. This facility has a database that records information about experimental data (*runs*) and the experimental logbook.

The application implements a Web site that allows users to search for Runs and logbook entries and to view the details of the database. The web site also allows users to update the logbook. There are some static HTML files in the client tier. They can supply search requests for users to submit to the servlet tier.

1 Introduction

The goal of my project was to enhance a Web site that displays information about scientific experiments and to add the ability to update that information. The previous application processes were implemented in C++ and a preliminary version in Java was available to me. For this project I completed the Java implementation using Java servlets, added new functionality, and created a system to allow the updating of some of the information. All of this work was completed and is being used in the production environment.

The Web site we developed (http://clasweb.jlab.org/clasonline) is located at the Department of Energy's Thomas Jefferson National Accelerator Facility (Jefferson Lab). This basic research laboratory was built to conduct experiments that probe the nucleus of the atom to learn more about the quark structure of matter. Our Web site is associated with Hall B, the most complex of the three experimental facilities at the lab.

The experimental data is organized into collections called *runs*. As part of collecting the data for experiments, computer programs extract primary information about each run and record that information in a database. Our Web site publishes the information about runs that is stored in the database.

Our Web site also publishes and manages the electronic logbook that experimenters use to record information about their activities. Experimenters create logbook entries from user interfaces within the experimental facility while data is collected. Other entries are created by data analysis software both during and after data collection. Still others are created by experimenters at remote locations.

The application has 3 tiers [5]: browser, Web server/Java servlet, and database server, as shown in Fig. 1. These tiers interact with each other to create the Web site. How does such an application work? The short answer is that the client, a web browser, sends an information request to a web server. The web server passes the request to the servlet. The servlet interacts with the back-end database. The database server returns contents to the servlet, which creates an HTML page that is returned to the browser.

The servlet works as a bridge between the client and the database tiers. It accepts a request from the browser client and checks the validity of the parameters. Then it determines what database information is required and sends appropriate SQL statements to the database server. It uses the information returned from the database server to create a response page in HTML. This result pages is transferred to the client.

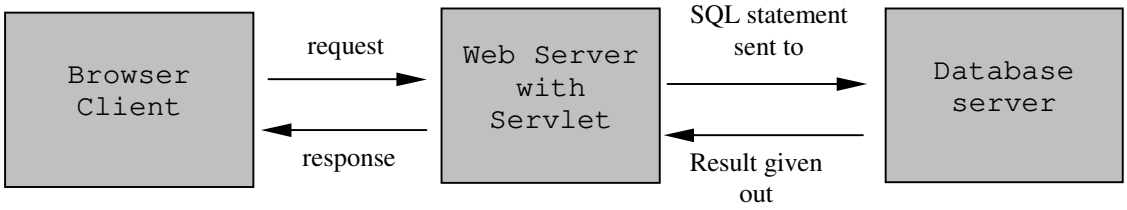

Figure 1. Three-tiered Web site

The application is one of the developments of our database research group. The research group director, Professor Riccardi, implemented the previous Web site software, selected

Java as the language for the new implementation, and created Java programs to create a new Web site. My job was to perfect the new Java implementation, modify it so that it used servlets instead of CGI scripts, and add an update capability for logbook entries.

Why was Java selected? Next paragraphs describe the main reasons.

There are a number of technologies available today that would allow us to accomplish such tasks. We could adopt CGI script, but this selection is filled with security risks, making it an unattractive solution to even experienced designers [1]. Active Server Page (ASP), a Microsoft technology designed to operate in conjunction with that company's Internet Information Server (IIS), is another solution, but it locks us into an operating system. A better solution is made possible by Java servlets and JDBC [2]. In this application, JDBC is used, which is described in Section 4, and Java servlets are enclosed, which is also described in later sections.

The Java servlet concept was first introduced in April 1997 [3]. Servlets are the server-side components that are independent on platforms. They may be thought of as applets for a Web server. They are almost identical to their CGI counterparts, and they can do anything that CGI can do. But servlets differ in several ways: they are generally easier to write than CGI scripts that are written in C++ or PERL, and they are much faster and much safer. The servlet tier in this application detects an action requested for and then passes the inner process into the associated sub-class that generates suitable result to be posted out or/and stored in. The practical process is given in Section 2.

A brief description for Java servlet's advantages can be given here. There are four advantage points for Java servlet [4]:

- Performance: Java Servlets do not require a new process for each request while CGI does. And if a server fails to load-balance or put a limit on the number of concurrent requests, it can easily be brought to its knees. The servlet `init( )` method allows programmers to perform resource-intensive operations common to all servlets invocations once at startup. In the servlet of my project, the `init( )` method builds up a database connection. This process can be done once only for the servlets requests. Therefore, the process performance speed is slowest for the first servlets invocation but sub-sequent invocations perform much faster.

- Portability: Java is platform independent so are servlets. The compiled servlet code can be moved from one machine to another without recompiling.

- Security: Servlets have the Java advantage -- memory access and strong typing violations are not possible.

- Flexibility: Although servlets are written in Java, their clients can be written in another programming language. Servlets can be written as clients to other servlets that are written in any programming language. For example, in our project, the database requests submitted via HTML form are processed by servlets using JDBC to connect an Oracle RDBMS.

Fig. 2 gives us the process architecture for the application developed in my project. The data flow and the logical relationships are depicted in this graph. There are 4 servlets in our application. One serves for Runs information, one for log entries information, one for a log entry update and one for a new log entry insert. This figure gives out the project architecture

by detailed graphs. The practical procedures for responses from these servlets are described in later sections.

In the client request interface tier, clients can view information, type information into forms, and submit requests to the Web server. The servlets will get information from the database and respond with client result interfaces.

Generally, the servlets need to connect to the database server, to create SQL statements, and then to send those statements to the database server. After a statement's execution, some entity results (ResultSet in the database server system) will be given to the servlets so that the responsible answer interface can be delivered to the browser.

Following depicts the interacting processes shown in Fig. 2.

(a) RunInfo Servlet

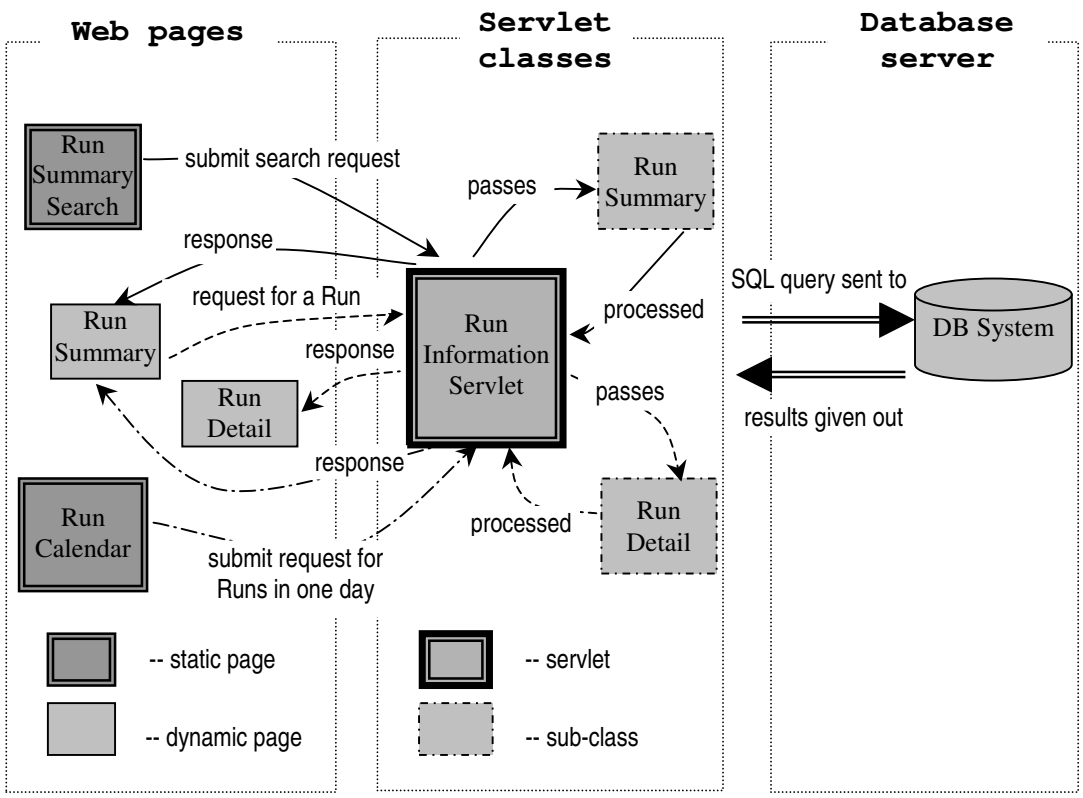

Figure 2. Four servlets in the 3-tiered information application: (a) Run Information, (b) Log Entry Information, (c) Log Entry Update and Insert

In the client tier, there are two kinds of Web pages: static and dynamic. The static pages are stored in files. The dynamic pages are created by the servlets in response to requests.

Each servlet class sets up the database connection, and passes the request to a suitable subclass for processing. Subclasses create SQL queries for database processing. The result returned from database is analyzed and a dynamic HTML page is returned to the browser.

Figure 2 (a) shows the Run Information Servlet. The static pages in the client tier are the Run Summary Search page and the Run Calendar pages. Clients can submit requests from these pages. The servlet receives the request and passes it to a related subclass. After the subclass interacts with the database tier, it will produce an appropriate response page.

The response pages are the dynamic ones shown in the client tier of Fig. 2(a). The response page for a Run Summary Search request is a Run Summary page. This page includes information about all runs that satisfy the search criteria. Links on a Run Summary page create requests that produce a Run Detail page. Response to requests on a Run Calendar page is also a Run Summary page.

Requests for Run Detail pages are processed by subclass `RunDetail`. Requests for Run Summary pages are processed by subclass `RunSummary`.

Figure 2 (b) shows the Log Entry Information Servlet. The static page in the client tier is the Log Entry Summary Search page. Clients submit search requests from this page. After the servlet receives the request, it will pass the request to its subclass `LogSummary`.

As shown in the client tier of Fig. 2 (b), the dynamic response pages are the Log Entry Summary page, the Log Entry Detail page, the Thread Headers page, and the Thread Summary page. From the Log Entry Summary page, clients can submit a log entry detail request. Response to this request is a Log Entry Detail page.

Two or more log entries can be bound in a thread. Threads are explained in more detail in Sections 7 and 8.

Clients can submit a request for a thread's summary from the dynamic Log Entry Detail page. Response to it is the Thread Summary page from which clients can also submit a request for the detail of a log entry in that thread.

From the Log Entry Summary page and the Thread Summary page, clients can also submit a request for thread header information. Response to such request is a Thread Headers page.

Requests for Log Entry Summary page, Thread Summary page and Thread Headers page are processed by the servlet's subclass `LogSummary`. Request for Log Entry Detail page is processed by subclass `LogEntryDetail`.

(b) Log Entry Information Servlet

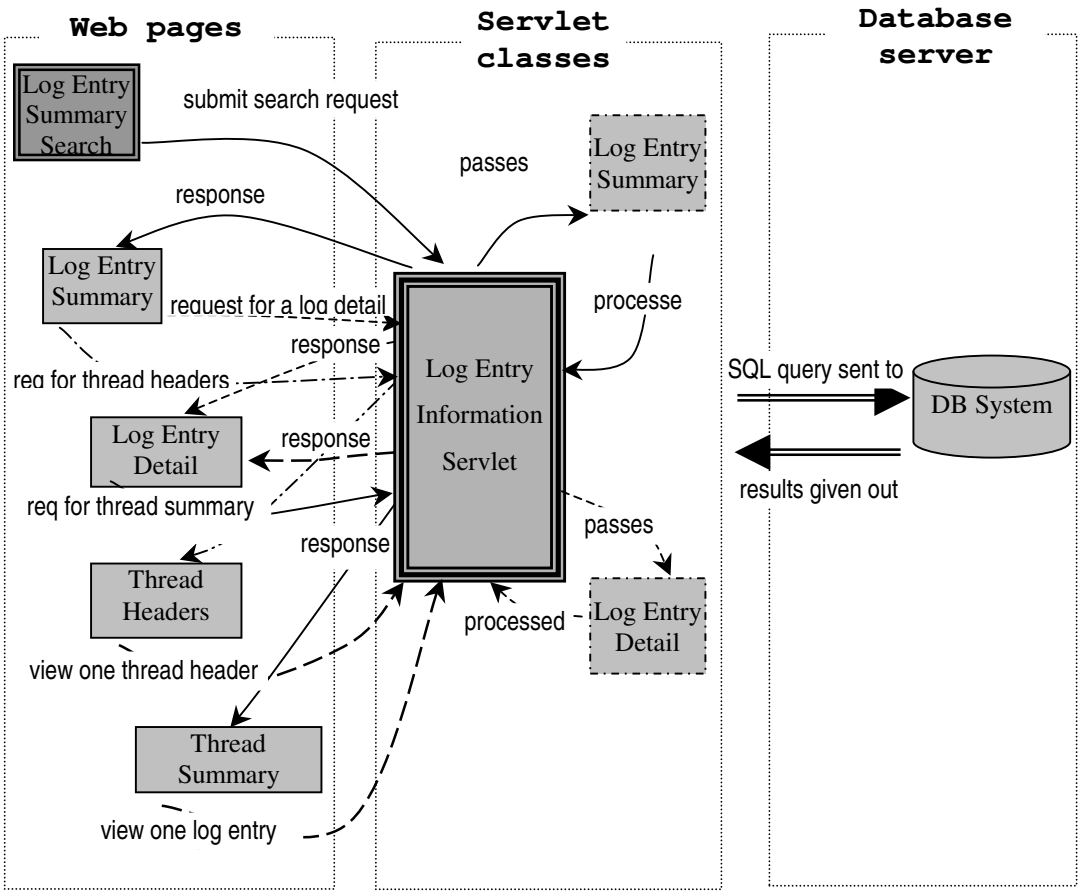

Figure 2 (b). Log Entry Information Servlet

Figure 2 (c) shows the Log Entry Insert and Update Servlets.

Clients can submit an update request from the dynamic Log Entry Detail page. A log entry may be related to a thread. The update process may include the changes on the log entry's thread. Particularly, a log entry update may create a new thread. More detail is in Section 9.

If the whole procedures for an update go on successfully, response to this update request is the dynamic Update Log Entry Detail page that displays the result of the update.

The update request is processed by the subclass, `LogEntryUpdate`.

Figure 2 (c) also shows the Log Entry Insert Servlet. There are three pages — one static and two dynamic — in the client tier, which allows clients to submit new log entry insert requests. The static page is the Log Entry Summary Search page and the dynamic pages are the Log Entry Summary page and the Log Entry Detail page.

(c) Log Entry Insert servlet

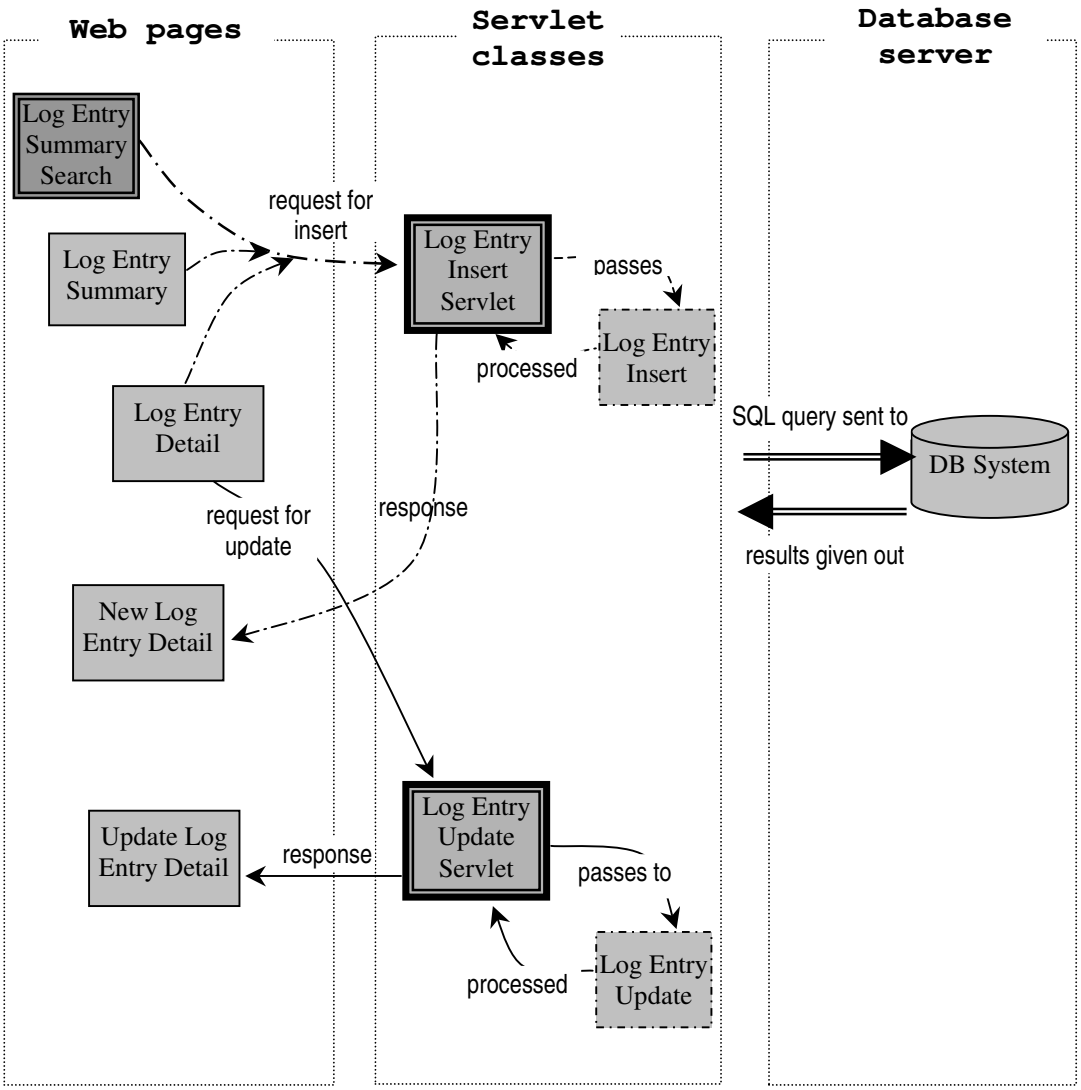

Figure 2 (c). Log Entry Insert and Update Servlets

For a new log entry, the updates to threads are similar to that in the update servlet. The insert servlet invokes a thread update method in the update servlet, if necessary.

If the whole procedures for an insert go on successfully, response to this insert request is the dynamic New Log Entry Detail page that displays the result of the update, which is shown in the client tier of Fig. 2 (c).

The insert request is processed by the subclass, `LogEntryInsert`.

2 Search Process

This section reports one relation from client tier to servlet tier, which is the search process. When a search request is sent to a servlet, what does that servlet do? Read the following two sub-sections. There are request interfaces to be given in this section. The response ones and database statement execution will be reported in other sections.

2.1 Runs Search

In Jefferson Lab, the computation system supplies many running jobs. For public clients, the Runs that worked in the system can be viewed through a search interface, which provides criteria for looking for a wanted field to be viewed Runs. This interface is shown in Fig. 3.

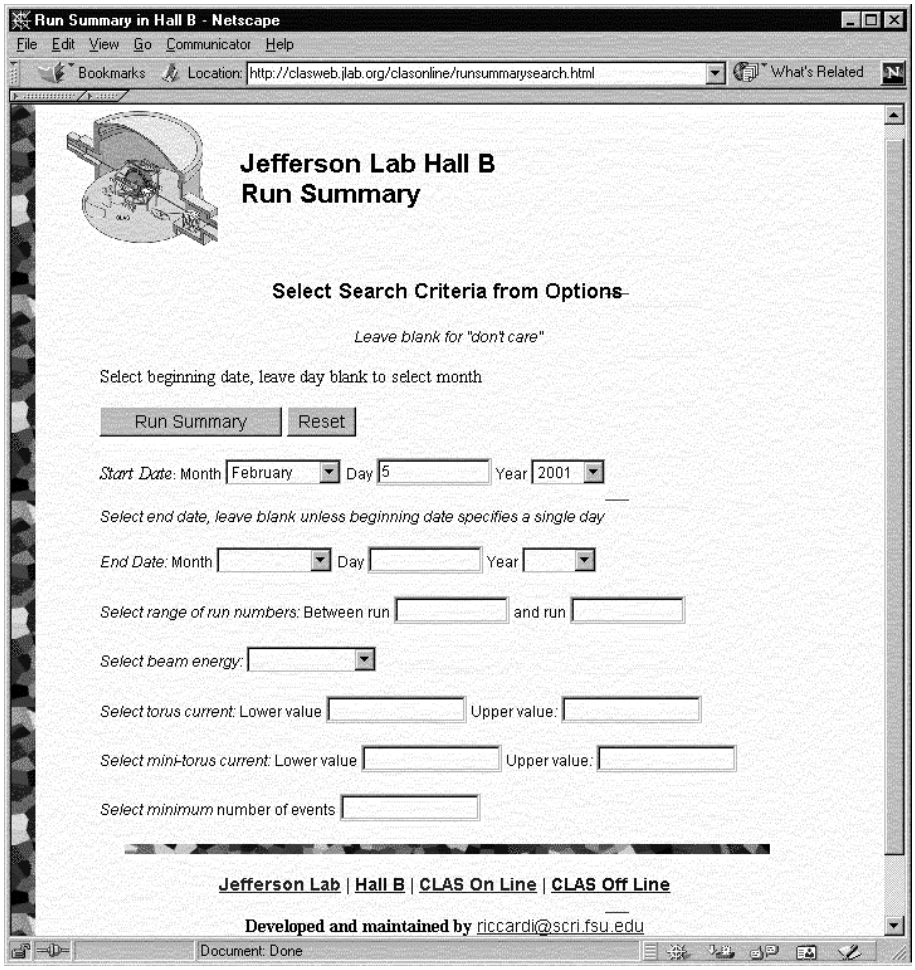

Figure 3. Run summary search interface

After the Run Information Servlet (`runinfo`) receives the Run search request, it will validate the search criteria and generate a where clause for a SQL query. For example, a client wants to find Runs on February 5, 2001. The search interface will send request to the `runinfo` servlet. The servlet will pass such request to its sub-class `RunSummary`.

`RunSummary` will validate the search criteria. If they are correct, a where clause will be generated as b.start_date >= '05-Feb-2001' AND b.start_date < '06-Feb-2001' for a transaction SQL query. The name b refers to the table ingres.run_log_begin in the database repository. The created query will be sent to the database system and executed. The result will be returned to the servlet. Then, the servlet will create the Run summary page as HTML for the client's request, which is in Fig. 4 below.

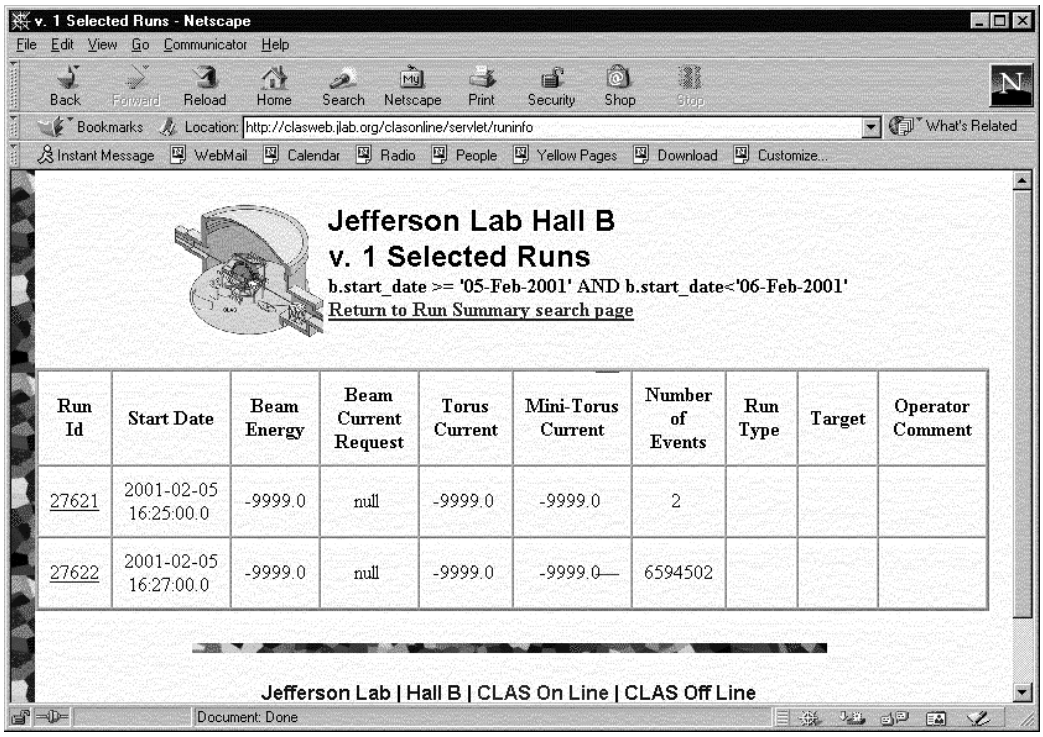

Figure 4. One Runs search result under condition: b.start_date >= '05-Feb-2001' AND b.start_date < '06-Feb-2001'

2.2 Log Entries Search

The information system also contains some legal log entries in security. For log entry summary information, the view action is public. So, the log information search interface is published for free entrance. This search page is shown Fig. 5.

As you can see, on this search page, besides the search form there is a request button for a new log entry insert. That process is not public but private in security, which will be reported in log entry insert Section.

The log search page submits the search request to Log Entry Information Servlet (`loginfo`) with `action=logsummary`. This servlet passes such action request to its sub-class `LogSummary`. `LogSummary` will also validate the search criteria to generate a where clause for a transaction query. The process is similar to the Runs search process except for the database tables. If seeking for log entries on the day July 5, 2001 and related to a Run 10338, the where clause for the transaction SQL query is

Search for Log Entries - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://clasweb.jlab.org/clasonline/loginfosearch.html
What's Related
Jefferson Lab Hall B
Search for Log Entries
Unresolved Logbook Entries
Select Search Criteria from Options
Search results returned in new window
Leave blank for "don't care"
New Log Entry
Latest Log Entry

ERROR: ioerror
OFFENDING COMMAND: image

STACK: