The Electronic Logbook for the Information Storage of ATLAS Experiment at LHC (ELisA)

A Corso Radu[1] **,G Lehmann Miotto**[2]**and L Magnoni**[2]
[1]University of California, Department of Physics and Astronomy, 4129 Frederick Reines Hall, Irvine, CA 92697-4575, USA
[2]CERN, PH Department, CH-1211 Geneve 23, Switzerland

E-mail: alina.radu@cern.ch

Abstract. A large experiment like ATLAS at LHC (CERN), with over three thousand members and a shift crew of 15 people running the experiment 24/7, needs an easy and reliable tool to gather all the information concerning the experiment development, installation, deployment and exploitation over its lifetime.
With the increasing number of users and the accumulation of stored information since the experiment start-up, the electronic logbook actually in use, ATLOG, started to show its limitations in terms of speed and usability. Its monolithic architecture makes the maintenance and implementation of new functionality a hard-to-almost-impossible process.
A new tool ELisA has been developed to replace the existing ATLOG. It is based on modern web technologies: the Spring framework using a Model-View-Controller architecture was chosen, thus helping building flexible and easy to maintain applications. The new tool implements all features of the old electronic logbook with increased performance and better graphics: it uses the same database back-end for portability reasons. In addition, several new requirements have been accommodated which could not be implemented in ATLOG.
This paper describes the architecture, implementation and performance of ELisA, with particular emphasis on the choices that allowed having a scalable and very fast system and on the aspects that could be re-used in different contexts to build a similar application.

1. **Introduction**
The ATLAS collaboration is composed of about 3000 scientific users from 38 countries who, together, run one of the largest high-energy physics experiments in the world.

An electronic logbook is used to record and share messages about ATLAS data taking activities by system operators, experts and automated services. Its general purpose is to make it easy for people to put information online in a chronological fashion, in the form of short, time-stamped text messages ("entries") with HTML mark-up for presentation, and optional file attachments (images, documents, etc).

The logbook was designed in its first version with a web front-end for message visualization and an Oracle back-end for archiving messages. Nevertheless, the original implementation (ATLOG based on ELOG [1]) showed limitations in scalability, performance and functionality (like update an entry) some due to the adopted web technologies.

ELisA, the new logbook implementation, is a modern web application that implements the Model-View-Controller (MVC) pattern and is based on the Spring [2] Web MVC framework. It provides an easy access to this information for other people through a Web interface, to browse entries, search, download files, and optionally add, update or reply on entries. Since the log messages are of interest for the whole ATLAS collaboration, the logbook tool has to meet strict requirements: high-availability and scalability, fast and effective messages manipulations (browsing, editing and visualization). In addition it privileges client-side processing for message visualization and uses the Ajax [3] technique to asynchronously retrieve data on client request. ELisA provides a web-based API for message manipulation.

2. Technology choices

The Spring MVC pattern introduces a well-defined structure in the ELisA architecture. The result is a flexible and loosely coupled web application achieved by separating the business logic, presentation logic and navigation logic. Models are responsible for encapsulating the application data. The Views render the response to the user with the help of the model object. Controllers (standard Spring beans) are responsible for receiving the request from the user and calling the back-end services.

Figure 1 shows the flow of request processing in the Spring MVC Framework. When a request is sent to the Spring MVC Framework the following sequence of events happens:

- The DispatcherServlet receives the request.
- The DispatcherServlet invokes the Controller associated with the request.
- The Controller processes the request by calling the appropriate service methods and returns a Model object to the DispatcherServlet. Model objects, mapped to the logbook entries, are retrieved from the Oracle database via the Spring Data and Hibernate tools.
- The DispatcherServlet passes the Model object to the View to render the result.
- The View with the help of the model data renders the result back to the user.

Client requests can come either from the web front-end or from a dedicated API (see section 4).

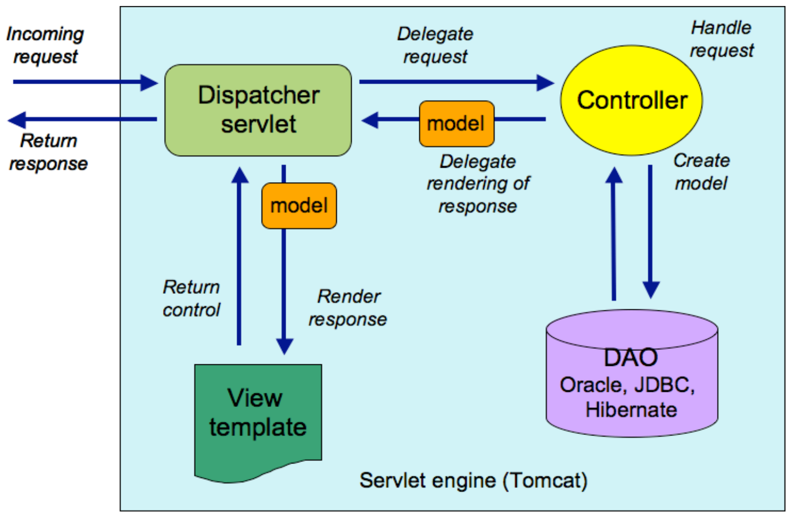


Figure 1. The flow of request processing in Spring Web MVC.

To improve the responsiveness level of ELisA we choose to use Ajax, a technique for creating interactive web applications. Ajax makes it possible to create web pages that are responsive by exchanging small amounts of data with the server behind the scenes in an asynchronous way, so that the entire web page does not have to be reloaded each time the user makes a change. This is meant to increase the web page's interactivity, speed, and usability. JavaScript Object Notation (JSON) is used as an alternative format for data interchange, instead of XML.

One drawback of using Ajax is that any user whose browser does not support JavaScript or XMLHttpRequest, or simply has this functionality disabled, will not be able to properly use pages that depend on it. In this case a particular alert message will warn the user about this issue.

These approaches add responsiveness to the user experience, reduce the load on the server and decouple the implementation from the back-end database. Moreover, using standard technologies and frameworks reduces the in-house developed code-base and thus facilitates software maintenance and evolution.

3. The web interface

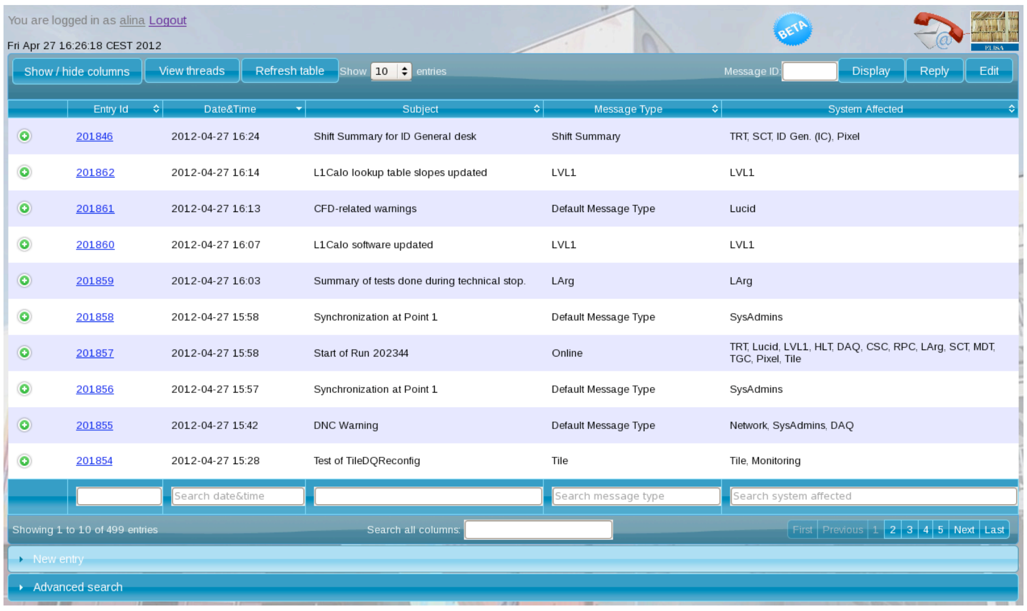
The ELisA web front-end is designed to provide an effective and responsive user interface to browse and edit logbook messages. Based on JavaServer Pages (JSP) technology [4], the front-end makes heavy use of JavaScript and a jQuery plug-in called DataTables [5] for client-side processing, data formatting and visualization.

Users are provided with functionalities to view and browse messages, insert new entries and reply or update existing ones (new functionality compared to the previous implementation of the logbook).

Users can configure the layout of the page by changing the page length size, the visibility of the columns in the entries table or the columns sorting criteria. Cookies are used in order to store page state after each change in drawing. If the user reloads the page, the table should remain as it was in terms of page length, filtering, pagination and sorting.

Users can perform fast searches on the data available on the client-side using on-the-fly filter functionality to select the data on one or more columns or searching across all columns using a dedicated user input field. An advanced search panel is provided to fill in a custom query form: searches based on the multiple properties of the message or time interval are possible. If several criteria are provided only entries that meet all criteria will be selected.

Figure 2 and figure 3 show two ELisA web interface screen shots that illustrate some of the functionality described in this section.

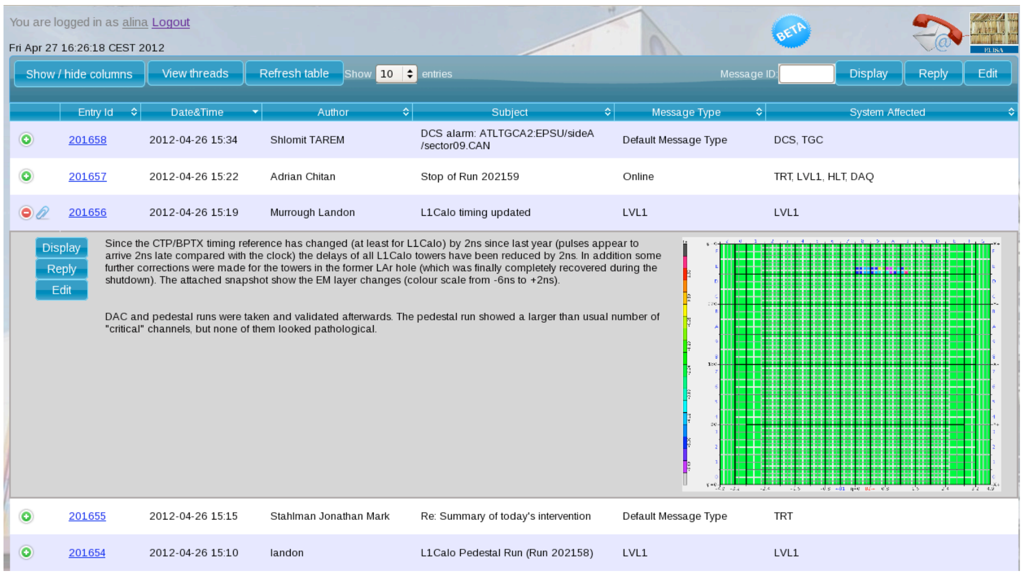

You are logged in as alina Logout
Fri Apr 27 16:26:18 CEST 2012

	Entry Id	Date&Time	Subject	Message Type	System Affected
⊕	201846	2012-04-27 16:24	Shift Summary for ID General desk	Shift Summary	TRT, SCT, ID Gen. (IC), Pixel
⊕	201862	2012-04-27 16:14	L1Calo lookup table slopes updated	LVL1	LVL1
⊕	201861	2012-04-27 16:13	CFD-related warnings	Default Message Type	Lucid
⊕	201860	2012-04-27 16:07	L1Calo software updated	LVL1	LVL1
⊕	201859	2012-04-27 16:03	Summary of tests done during technical stop.	LArg	LArg
⊕	201858	2012-04-27 15:58	Synchronization at Point 1	Default Message Type	SysAdmins
⊕	201857	2012-04-27 15:58	Start of Run 202344	Online	TRT, Lucid, LVL1, HLT, DAQ, CSC, RPC, LArg, SCT, MDT, TGC, Pixel, Tile
⊕	201856	2012-04-27 15:57	Synchronization at Point 1	Default Message Type	SysAdmins
⊕	201855	2012-04-27 15:42	DNC Warning	Default Message Type	Network, SysAdmins, DAQ
⊕	201854	2012-04-27 15:28	Test of TileDQReconfig	Tile	Tile, Monitoring

Showing 1 to 10 of 499 entries

Figure 2. ELisA web interface: main page.

An asynchronous communication with the ELisA back-end is implemented using Ajax: basic entry properties are pre-loaded on the client-side and the detailed entry content (textual content and optional attachments) is retrieved on user request only. This allows for a fast and effective user experience, which is one of the main reasons of the ELisA project success.



You are logged in as alina Logout

Fri Apr 27 16:26:18 CEST 2012

Show / hide columns | View threads | Refresh table | Show 10 entries | Message ID: | Display | Reply | Edit

	Entry Id	Date&Time	Author	Subject	Message Type	System Affected
⊕	201658	2012-04-26 15:34	Shlomit TAREM	DCS alarm: ATLTGCA2:EPSU/sideA /sector09.CAN	Default Message Type	DCS, TGC
⊕	201657	2012-04-26 15:22	Adrian Chitan	Stop of Run 202159	Online	TRT, LVL1, HLT, DAQ
⊖	201656	2012-04-26 15:19	Murrough Landon	L1Calo timing updated	LVL1	LVL1

Since the CTP/BPTX timing reference has changed (at least for L1Calo) by 2ns since last year (pulses appear to arrive 2ns late compared with the clock) the delays of all L1Calo towers have been reduced by 2ns. In addition some further corrections were made for the towers in the former LAr hole (which was finally completely recovered during the shutdown). The attached snapshot show the EM layer changes (colour scale from -6ns to +2ns).

DAC and pedestal runs were taken and validated afterwards. The pedestal run showed a larger than usual number of "critical" channels, but none of them looked pathological.

	Entry Id	Date&Time	Author	Subject	Message Type	System Affected
⊕	201655	2012-04-26 15:15	Stahlman Jonathan Mark	Re: Summary of today's intervention	Default Message Type	TRT
⊕	201654	2012-04-26 15:10	landon	L1Calo Pedestal Run (Run 202158)	LVL1	LVL1

Figure 3. ELisA web interface: entry expanded view.

4. The web API

An HTTP-based REST API is provided for client applications, such as automated scripts to access and modify logbook messages. Following the REST architectural principles [6], a client makes a HTTP request using the standard verbs (GET/PUT/POST/DELETE) to a structured URL. Operations such as creating a new entry, editing existing ones or getting a list of entries with specified criteria are supported in this way.

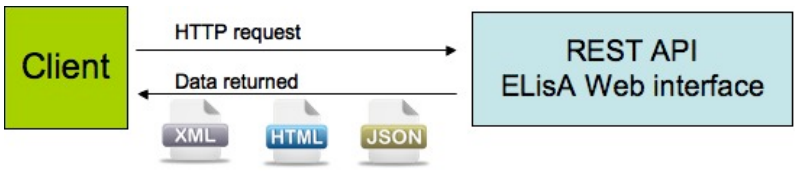


Figure 4. Different possible representations of requested resources.

The client obtains (or sends) a representation of the requested resource in one of several possible formats, such as JSON, XML or HTML, as shown in figure 4. This standard and lightweight approach is fully integrated with ELisA's MVC design and decouples the server implementation from the client technologies.

5. Deployment

The ELisA back-end server, hosted by the Apache Tomcat container, is deployed on two dedicated server machines, one used for production and the other reserved for fail-over, both installed on the Atlas Control Network.

In order to use ELisA a user needs to be authenticated first. This procedure is based on CERN Shibboleth Single-Sign-On for usage on the General Public Network and on plain LDAP for usage inside the restricted experiment network.

6. Performance

Figure 5 shows the comparison of the front-end loading time for the old logbook implementation (ATLOG) and for ELisA, meaning the time to retrieve a number of entries and to render them on the main page of the logbook (for ELisA main page see figure 2).

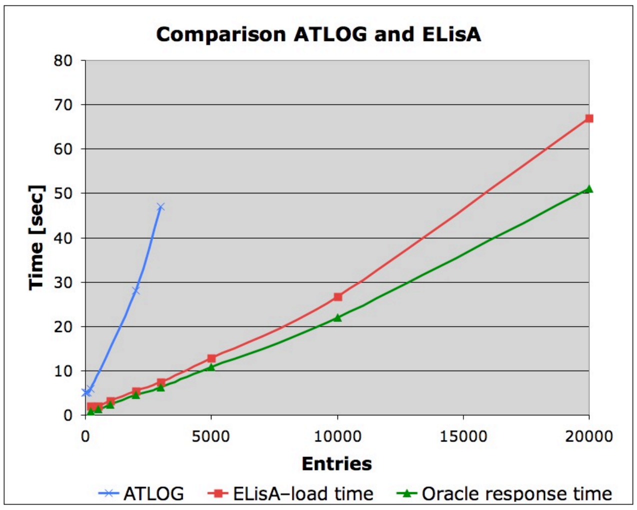


Figure 5. Load time comparison between ELisA and ATLOG.

The improvement of ELisA compared with the old ATLOG implementation is clearly visible in the plot. The new implementation scales well with the number of messages retrieved, and it adds a minor time overhead on top of the Oracle query-response time. The database schema has been preserved for backward compatibility, but it will be subject to revision and optimization in the future to further improve the data retrieval performance.

7. Conclusions

ELisA has been used by ATLAS in production since the beginning of data taking in 2012 and has been positively received by many users. Client-side processing allows for a faster and more effective user experience. The adoption of an MVC-driven architecture has allowed focusing code development on specific features of the logbook, while profiting from the reliability of established third-party technologies such as the Spring framework.

The implementation of a REST API provides access to the logbook features to clients and services beyond the provided web front-end.

8. References

[1] ELOG: http://midas.psi.ch/elog/
[2] Spring Framework: http://static.springsource.org/
[3] Zakas N, McPeak J and Fawcett J 2007 *Professional Ajax, 2nd Edition (Programmer to Programmer)* (Wrox)
[4] Bergsten H 2003 *JavaServer Pages (3rd edition)* (O'Reilly Media) ISBN 978-0-596-00563-4
[5] DataTables: http://www.datatables.net/
[6] RESTful web services: https://www.ibm.com/developerworks/webservices/library/ws-restful/

Acknowledgments
We would like to thank the ATLAS TDAQ Sys-admin team for their work and support during the deployment process of ELisA on the ATLAS web servers. As well many thanks to ATLAS Databases group for their assistance and advice on Oracle database related issues.