An environment for managing Web database applications

José Paulo Leal

Technical Report Series: DCC-98-5

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823 4150 Porto, Portugal

Tel: +351+2+6078830 – Fax: +351+2+6003654

http://www.ncc.up.pt/fcup/DCC/Pubs/treports.html

An environment for managing Web database applications

José Paulo Leal

zp@ncc.up.pt
DCC - FC & LIACC, Universidade do Porto
R. do Campo Alegre, 823 4150 Porto – Portugal

October 1998

1 Introduction

This paper describes Aleph, an environment for developing distributed applications on the World Wide Web (WWW). Its main design goal is to assist non-programmers in the creation of WWW applications involving the cooperative management of small databases. The application programmer is expected to have only a minimal knowledge about databases to structure the application and no programming skills are required to define queries or to create functionalities. The system provides tools to format graphical interfaces using an extension of HTML[RH98] that is compatible with existing HTML formatting utilities.

A typical application developed with Aleph manages a database of a few thousand records, containing text and images in formats supported by the WWW. The records can be inserted by any Internet use and, in general, can only be edited by the user that produced it. The updated data can either must be immediately accessible to all users, or require validation by a privileged user. The navigation in the database uses hypertext links and may be based on a hierarchical structure.

Aleph may be used in any computer connected to the Internet. Like the final applications developed in Aleph, the system itself is a WWW application, thus it can be operated from any platform. The Aleph server is implemented in version 8 of tcl[Ous94]. This language was chosen for its stability, reduced development cycle and availability in the most computing platforms. The environment is already in a production stage and is being used by professional designers in a Web designing company.

2 Background

The WWW is arguably the best environment for developing distributed applications. This Internet service provides a widely available set of tools for managing graphical interfaces and communicating with remote systems. These tools, namely the WWW browser and the HTTP[NG98] server, enable the integration of hypertext navigation and multimedia contents in the creation of distributed applications.

Nevertheless, the WWW has also features that actually difficult the implementation of distributed applications. The statelessness of the HTTP transference protocol is a necessary feature to support distributed hypertext navigation but is generally regarded as the major source of problems when implementing distributed applications[Per94]. The statelessness of HTTP is reflected in the fact that user interaction in WWW is based on a sequence of independent request-response cycles. In other words, each browser request to a server is independent from previous requests even if made to the same server. Two problems created by the statelessness of HTTP have to be considered when creating a distributed application:

Persistence of database connections A working session with a Database Management System (DBMS) in general involves several queries. The resources needed to handle these queries as created on the connection to the DBMS. Due to the statelessness of HTTP the connection has to be repeated for each query.

Management of user interfaces The interface of a WWW application is formated using HTML pages. To update the interface to a new state the application requests a new page to replace (part of) the previous interface. The new interface depends not only on the previous page but also on all previous interaction.

Languages such as Java[jav, AG96] and JavaScript[Fla97, net], available in recent versions of the major WWW browsers, are often considered as implementation languages for distributed applications in the WWW. These languages have protocols, such as JDBC and LiveConnect, for communicating directly with databases, and can be used for managing the users interface, thus solving both problems resulting from the statelessness.

Browser languages are obvious choices for implementing distributed applications for Intranets. In this scenario the programmer may assume that a specific version of a WWW browser with the specific versions of each required package is installed in the user's machine. In an Internet scenario the programmer should make very few assumptions about the browser. In their rush to dominate the market, the two major browser vendors created a large set of incompatible features. Hence, a distributed application that relies on features of a particular version of JavaScript or JDK (the Java's API distributed with the browser) may not run in most browsers. This is particularly true in Portugal were 50% of the browser are Modzilla 2.* compatible[1], meaning that 50% of the Portuguese browsers support only Java with JDK 1.0 and JavaScript 1.0 without respectively JDBC and LiveConnect for connecting with remote databases. Moreover, there are two distinct browsers with this signature, Netscape navigator 2.* and Internet Explorer 3.*, that are not entirely compatible with each other.

The lack of a standard in browser languages forces most WWW application to work around the statelessness problem using standard HTML and the HTTP features, such as hidden fields, URLs parameters and cookies. Writing distributed application using this approach is very simple but it is also tedious and error-prone, and the resulting code is in general difficult to maintain since is it intertwined with HTML layout.

3 The Aleph System

The reseach in this project focused on the development tools to implement distributed Web applications without requiring computation on the client side. **Aleph**, the resulting system is an environment to assist non-programmers in the creation of Web applications involving the cooperative management of small databases. The main features of the Aleph system are the following:

WWW oriented database management system The Aleph database management system (DBMS) was specially designed for WWW applications. The DBMS is integrated in the application's server thus solving the problem of persistence in database connections. Aleph databases are collections of linked tables. The links between tables lack the flexibility (and complexity) of relational systems, but are powerful enough to create complex data structures such as hierarchical structures. Aleph supports only the basic data types that can be integrated in HTML forms (text and images) and has no size restriction. The simplicity of Aleph databases allow restructuring the application's database even with the application on-line.

Automation of frequent operations The Aleph system implements a set of operations that can be used to define interactively most, if not all, commands of the final application. Operations are defined simply by selecting a type and a name in the Aleph monitor. The new operations inherits the properties defined interactively in Aleph's monitor. These properties include database and table names, search criteria and sorting keys. The operation is encoded in a URL that can be

[1] This figure was measured in the AEIOU (www.aeiou.pt) a Portuguese search engine.

integrated anywhere on the final application users interface. Aleph operations are sequences of requests returning the following interface state. This approach solves the problem of managing graphical interfaces within the limitations of the HTTP without requiring any programming.

Separation of functionality and layout The graphical layout of users interfaces in Aleph applications is formatted using an extension of the standard HTML language. This approach doesn't require embedding programming code or SQL statements as other systems do[LGS, web]. The definition of a new operation generates a skeleton of each page layout that can be refined using any standard HTML utility. Aleph includes an editor for fine tuning that generates the final page with actual data when the layout is updated. To facilitate debugging any error occurring in this stage is reported within Aleph and not on HTTP log files.

In the remainder of this section is described the structure of Aleph and explained with more detail how operations are defined and graphically formated. The section ends with a description of the client-server communication methods supported by Aleph.

3.1 System's structure

The Aleph system follows a client-server architecture, similar to those of the final applications it generates. Aleph's server contain the database management system and the configuration of the operations included in the final applications. The Aleph server processes also the requests for operations included in the final application.

Final applications are created and managed using Aleph's front-end – the **monitor**. The monitor (see Figure 2) is a distributed application that displays its interface in a WWW browser and communicates with Aleph's server to configure the database and the application. The monitor's interface presents a workspace to operate interactively with the final applications database, to define its operations and its graphical interface.

The interface of the monitor presents a workspace to operate with the applications database. Using graphical interaction the application manager can execute commands to manage the database structure and contents, execute queries and sort the results.

3.2 Operations

An Aleph application is assembled using a set of operations over database tables. An operation is a sequence of request-reply cycles trigger by an initial request. For instance, a query is initiated by requesting a form, the form is linked to a list of matching records and each record in turn may be linked to a view of the record's content. Aleph supports several types of operation that systemize the usual funcionalities of a distributed application.

The current Aleph implementation supports four operation types, i.e. four different sequences of page requests, and each request generates a page of a one of four basic types. Table 1 presents the sequences of pages for each operation type. Each line corresponds to an operation and the pages are generated in the order they appear in the table, from left to right.

Operation / Page	Authentication	Form	List	Record
List			x	x
Search		x		x
Insert		x		x
Edit	x	x		x

Table 1: Aleph operations and page types

The search operation illustrates how the different page types are connected. A form is generated, and after filled and submitted by the user, generates a list of records matching the criteria. The records in the listing can be linked to a page where the record is presented in detail, possibly with more fields.

Each Aleph operation is identified by a **name**. Hence, several operation of the same type, for the same table, may coexist in a single application. Aleph associates to the operation's name the context that was active when the operation was defined. In particular Aleph records the database, the table, the search criteria, and the sorting keys, associated with the operation's name. The operation's context is used in each request for an operation's page. The request is encoded in a URL whose prefix locates Aleph base directory, and whose suffix indicates the type of page and name of operation. For instance, assuming that Aleph resides at `http://www.ncc.up.pt/ zp/aleph` and has a listing named `ex`, then the listing is invoked by the following URL

http://www.ncc.up.pt/ zp/aleph/lista?ex

3.3 Page layout

The layout of the pages is the only part of the execution context that may be changed after the creation of an operation. At creation time a layout is generated for each page included in the operation. This default layout is enough for visualizing and can be used as a skeleton for the final page layout.

The page layout is a template of an HTML formatted page. In the template the field values are referred by the field name preceded by a dollar sign. For instance, if the table includes the field `NAME` then the layout of a record may include a reference to `$NAME` that is replaced by the actual value of the field each time it is used to present a record.

In the case of a listing the layout is a bit more complex since it requires a template with several parts. In particular there are two template parts for the header and footer and a separate template part to format each record in the listing. These three parts are included in a single file separated in containers defined by the special element `!part`. This element must be complemented with the attribute `ext` to indicate the type of part it is defining. These tags defined this element and attribute follow the HTML syntax but are ignored by the WWW browsers that interpret them as comments. Inside the containers defined by these tags is placed the layout of each part, including HTML tags, text and references to field values. The layout of a listing has the following skeleton

```
<!part ext="cab">
... layout of header ..
<!part>
<!part ext="lin">
... layout of each line ...
<!part>
<!part ext="rod">
... layout of footer ...
<!part>
```

Figure 1: Skeleton of a listing layout

The Aleph monitor includes a table with all defined operations, their types, databases, tables and associated pages. From this table the application monitor can select the existing page layouts and reformat them. The monitor includes a layout editor where the current layout can be changed and the resulting page vizualized using the values currently defined in the database. Alternatively, the layout can be downloaded, formated in a generic HTML editor and them uploaded again to the Aleph.

3.4 Client-server communication

As mentioned before, WWW applications are structured as a sequence of request-response cycles from a WWW browser to a remote HTTP server. The requests are initiated by HTML formated pages, presented in a WWW browser, and the responses generate a new HTML formated page. The Aleph system provides two methods for client-server communication:

- Indirect communication using the CGI protocol
- Direct communication using HTTP protocol

The CGI protocol allows the invocation of remote programs in the request-response cycle. The request is directed to a HTTP server that launches the execution on an external process that does the actual processing. This approach as the advantage of being very simple to implement and maintain using any standard HTTP server.

For applications with larger databases the CGI approach is not very efficient since it requires loading the entire database for each request. For this situations the Aleph system provides an HTTP server that loads the database and handles the HTTP request-response cycle, without requiring the execution of an external process.

4 Evaluation and Future Work

Aleph is being currently used in the development of distributed applications integrated in WWW sites of several companies and institutions. The system is being used by designers with no programming skills. After defining the application structure, implementing its functionalities takes sometimes less than a person's working day. This contrasts with the several days taken by a programmer to implement and maintain a similar application from scratch.

Most of the developing effort using Aleph is concentrated in formatting the graphical interface, and benefits from several features that simplify the integration of HTML layout in application. The most important improvement results from not requiring coordinating the work of people with different skills, namely the graphical designer and the programmer.

The feedback given by the people using Aleph has been very important to identify missing features in the current implementation. These features include new operations, such as free text search, and improvements in the database management system, such as computed fields.

4.1 References

[AG96] Ken Arnold and James Gosling. *The Java Programming Language.* Addison Wesley, 1996.

[Fla97] David Flanagan. *JavaScript: The Definitive Guide, 2nd Edition.* O'Reilly & Associates, 1997.

[jav] Javasoft. http://www.javasoft.com/.

[LGS] Rasmus Lerdorf, Andi Gutmans, and Zeev Suraski. Php hypertext preprocessor. http://www.php.net/.

[net] Javascript: Netscape documentation. http://developer.netscape.com/docs/.

[NG98] Henrik Frystyk Nielsen and Jim Gettys. Http - hypertext transfer protocol. http://www.w3.org/Protocols/, 1998.

[Ous94] J.K. Ousterhout. *TCL and the TK Toolkit.* Addison Wesley, Reading, MA, 1994.

[Per94] Louis Perrochon. Translation servers: Gateways between stateless and stateful information systems. In *Proceedings of the Network Service Conference*, London, November 1994.

[RH98] Dave Raggett and Arnaud Le Hor. Hypertext markup language. http://www.w3.org/MarkUp/, 1998.

[web] Oracle webdb. http://oracle.com/products/tools/webdb/.

Figure 2: Aleph monitor