Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

WDB—A Web Interface to Sybase

B. F. Rasmussen

Space Telescope-European Coordinating Facility, Karl-Schwarzschild-Strasse 2, D-85748 Garching bei München, Germany

Abstract. WDB is a software toolset that tremendously simplifies the integration of Sybase databases into the World Wide Web. WDB lets you provide WWW access to the contents of Sybase databases without writing a single line of code, yet provides the flexibility to easily add new capabilities as needed. This article describes the technical implications of implementing such an Interface. The tool itself, and full documentation is available on the WWW at theWDB Home Page[1].

1. Introduction

To run WDB, all that is needed is the WDB script (written in `sybperl`) and a set of high-level *form definition files*, each describing a different view on the database. WDB automatically creates HTML forms, *on-the-fly*, to allow users to query the database, and given these query constraints, it will query the database and present the result to the user. WDB even comes with a utility to automatically extract information about a table from the database and create a working template *form definition file*.

A number of conversions are possible on the data coming from the database before they are shown to users: formatting of coordinates into hours, minutes, and seconds; formatting dates; etc. The most noticeable feature is the possibility to convert data from the database into hypertext links—and as it is possible through WDB to access any database element directly via a Web URL—the entire database can then be turned into a huge hypertext system. These hypertext links can be links to other elements in the database, thereby providing a simple way of jumping between related information. There could also be links to other documents on the Web, providing easily integration between data in the database and related documents on the Web. One could also have links to other databases with a WDB or similar interface—providing a simple mechanism for cross-database links.

---

[1]http://arch-http.hq.eso.org/bfrasmus/wdb/wdb.html

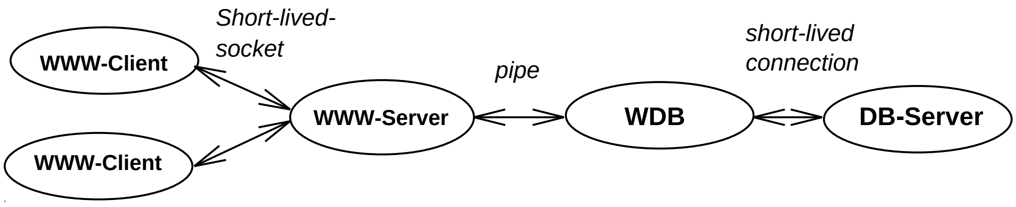

Figure 1. A WDB Connection.

2. Architecture

WDB is implemented as a CGI script which can generate forms and send them to the user, retrieve the contents of the form, process it and return the results to the user. The forms and description of how the individual fields in the form links to fields in the database are defined using *Form Definition Files* (*fdf's*) located on the server.

Due to the nature of a WWW connection, the WDB system has to be designed differently from a normal database application (see Figure 1). As there are no long-lived connections, the server has no notion of what has been going on earlier in a session. The WDB process is exited immediately after the results are send back to the user. This means that all information needed to perform a given action has to be sent from the client each time. Of course, the client will have this information from the last result it received from the server.

The way this information is sent back and forth between the WWW client and the server is by the URL. It is a standard for specifying a piece of information on the Internet, and is composed of a protocol name (ftp, http, gopher), the name of the server, and a path on that server pointing to the piece of information needed. This can be a file or, on an http server, a CGI script such as WDB. The trick is that if there is any extra information in the URL after the path, then it is passed to the script as what is called PATHINFO.

(ex. `http://arch-http.hq.eso.org/cgi-bin/wdb/`*PATHINFO*)

2.1. Flow of Control in WDB.

The actual flow of control is better illustrated in Figure 2. What happens is the following :

1. One "clicks" on a URL leading to the WDB script. The URL contains extra information in the path (PATHINFO), specifying what information in the database the user would like to query. This is typically obtained from a normal hypertext page with a menu of available query forms.

2. The server receives the URL request, starts the WDB script and passes the extra information in PATHINFO, actually the name of the *Form Definition File (fdf)* to use, and the word "form".

3. WDB reads the *fdf* and generates an HTML form according to the specifications in the *fdf*. The form is returned to the user and WDB exits.

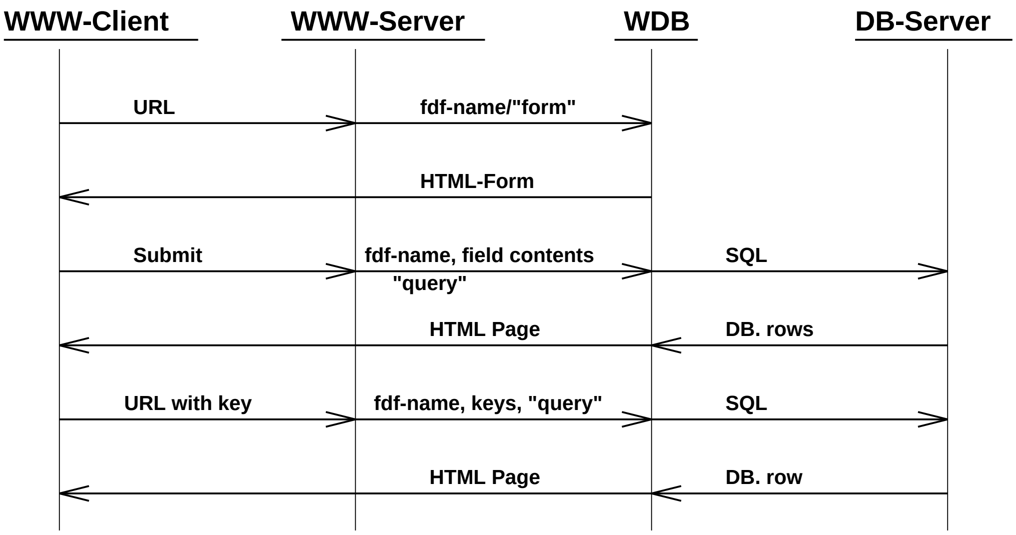

Figure 2. Flow of Control.

4. The user fills out the form with the query constraints and presses the "Submit" button. This sends the WDB URL (with PATHINFO) plus the contents of the form (encoded) to the WWW Server.

5. The server receives the URL, starts the WDB script and passes the extra information in PATHINFO, plus the form content.

6. WDB parses the PATHINFO which in this case is the name of the *fdf* plus the word "query". This tells WDB to construct a SQL query from the information in the *fdf* file and the user's constraints.

7. The SQL query is send to the database server.

8. The database server processes the query and returns a number of rows.

9. WDB retrieves the rows from the database server and after having converted the data according to the specifications in the *fdf*, constructs an HTML page with the result. If more than one row was returned, a tabular output format is chosen, allowing the user to select a row, via a hypertext link, to get all the details.

10. The HTML page is returned to the user, and WDB exits.

11. The user clicks on a hypertext link for a row. This link contains the WDB URL with the *fdf* name, the word "query" and the key(s) for the current row in the PATHINFO.

12. The WWW server receives the URL, starts the WDB script and passes it the extra information in PATHINFO.

13. WDB constructs a new database query, this time using the key(s) instead of the form contents.

14. The database processes the query and returns the result.
15. WDB retrieves the row from the database server, converts the data according to the specifications in the *fdf*, and this time, as only one row is returned, constructs a HTML page with all the details. The HTML page is returned to the user and WDB exits.

3. Conclusions

WDB has proven very successful—mainly because of its simplicity and the ease with which it can be extended. One of the goals of WDB was to provide an easy, yet flexible, way to make database information available. In the simple case, no coding should be necessary. However, if special behavior is needed, it should be possible to add it *without changing the core system*. Also, if some special behavior has been defined for one field in a form, it should be easy to reuse this in other forms.

This has been achieved by allowing extensions to be added in the individual *Form Definition Files* using the powerful Perl language. This makes local customization a lot easier than with most existing WWW/Database gateways.

WDB already has a large user base. At the time of writing, more than 30 sites world-wide have installed it. An e-mail mailing list is maintained for announcements of new releases. To be added to this list, send e-mail to the author (*bfrasmus@eso.org*).

References

McCool, Rob 1993, The Common Gateway Interface[2], National Center for Supercomputing Applications (Champaign, University of Illinois)

McCool, Rob 1994, NCSA httpd Overview[3], National Center for Supercomputing Applications (Champaign, University of Illinois)

Brenner, Steven E. 1994, CGI Form Handling in Perl[4] (Cambridge, University of Cambridge)

[2]http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

[3]http://hoohoo.ncsa.uiuc.edu/docs/Overview.html

[4]http://www.bio.cam.ac.uk/web/form.html