Design and Implementation of a Publication Database for the Vienna University of Technology

Karl Riedling
Institute of Industrial Electronics and Material Science, TU Wien, A-1040 Vienna
karl.riedling@tuwien.ac.at

Abstract:

Initially for the internal use of the EE faculty of the Vienna University of Technology and as a tool for the evaluation of the scientific output of the faculty's institutes, the author has created a database for publications. The first prototype based on *Microsoft Access* was in due course replaced by a Web-based solution with a LAMP server concept. Because the Web software met the expectations of the university authorities, it was implemented university-wide in mid-2002, and since then provides all publication-related evaluation data of the university. This presentation will describe the design of the Publication Database and will give some information on technical and organisational problems encountered during its implementation at the university, particularly aspects of how to improve user acceptance.

Introduction

The scientific community generally measures the quality of scientific work by the resulting published output. However, a simple count of the publications of researchers or institutes is hardly an accurate representation of this output due to the wide range of what different people consider a scientific publication, and because a simple publication count supplied by the researchers often cannot be fully verified. There are official databases of recognized publications for some areas but not for all, notably Electrical Engineering. With the aim to provide reliable publication data for the EE faculty of the Vienna University of Technology that are suitable for resolving all conceivable pertinent queries, the author has devised the Publication Database. Since a quick solution was required, the author chose *Microsoft Access* for the prototype version of this database. The *Access* prototype consisted of two modules, a GUI front-end with a number of VB script modules as a user interface that could easily be upgraded, and a back-end that held the publication data. This database became operational after only a few months of development; after a few more months of test operation at the author's institute it was introduced faculty-wide in late 1999, first with separate copies of the Database for each of the about fifteen institutes of the faculty, later with one common server-based installation. The severe drawbacks of the *Access* application soon showed:

- Bad acceptance by institutes that found themselves forced to set up a computer under *Windows* and install *Access*;
- Compatibility problems, since it was not possible to simultaneously run the Database on different versions of *Access* or upgrade it automatically to a newer version;

- Frequently damaged data sets due to computers crashing with the *Access* Publication Database open;
- Consequently, an excessive need for database maintenance;
- No reasonable possibility for real-time queries by Web servers;
- And over all the lack of any serious security because even looking up data on an *Access* database requires write permission to the database file.

These considerations and the prospect of a much more powerful system led to the development of a Web-based database solution with a LAMP (Linux – Apache – MySQL – PHP) approach. Based on the concept of and the experience gathered with the *Access* prototype, and under the author's supervision, a group of four students developed the code of the Web Database, which took more than one year due to the complexity of the task involved. Hence, the Web version became available only in mid-2001, almost two years after the *Access* prototype had been ready for use. After not less than 13 version releases of the *Access* Database, the Web Database took over the data and the tasks of the prototype Publication Database.

Since mid-2001, the author is the (practically) sole person in charge of the Web Publication Database; he has added a wealth of additional functions and improvements meanwhile. The Database has grown from 25 tables in its *Access* version to 30 in the early Web releases, and comprises 47 tables in its current 19th release. According to a recognised tool for software assessment, David A. Wheeler's "*SLOCCount*" [1], the Web database consists of 18,260 lines of PHP program code today, which corresponds to more than 4.5 person-years of development effort (and development costs of 570,000 US-$). (*SLOCCount* cannot count the lines of HTML code in the Web database, whose number lies in a similar order of magnitude.)

Although the *Access* prototype version already provided much more functionality than required for evaluation purposes, we could significantly increase the "added value" of the Database in the Web version. Because the software met the expectations of the university authorities, it has finally been implemented university-wide in mid-2002, and since then provides all publication-related evaluation data of the university.

The Concept of the Publication Database

The design of a system like the Publication Database has to take into account two possibly conflicting requirements. Information in the Database has to be as complete and detailed as possible to allow for all conceivable queries which may not only result in a simple count of publications but should also take into account the quality of the publications (or, easier to maintain, of the media where the publications appeared). On the other hand, entries into the Database are often made by persons not familiar with the detailed aspects of bibliography (e.g., by an institute secretary). This precludes a "full-blown" bibliographic system and demands a flexible approach where only the fields essential for identifying and verifying a publication need filling in, while optional fields are available for additional information such as abstracts, identification numbers in internationally recognised publication collections for the particular field, or a link to an electronic version of the publication.

[1] http://www.dwheeler.com/sloccount/

In general, instruments that only serve the purpose to collect statistical data are not well accepted. In order to improve the acceptance of the Publication Database, it has to provide sufficient “added value” to its users. This entails that, e.g., everybody must be able to extract their own publication lists or have them created dynamically in any perceivable format for use on a website, and that external users can freely search for information in the Database. In fact, the Publication Database must serve as a knowledge base as well.

Both operations – the collection of evaluation data and the operation as a knowledge base with the possibility to search for information – imply a genuine database structure, where each item of a publication entry is located in a separate field of a database table. Obviously, there are relations between the data fields; hence, a relational database is required. Several authors affiliated to different institutes of the faculty may jointly have written a publication, which is supposed to appear in the publication lists of each of its authors, and of each of the groups and institutes to which its authors belong. This implies that the names of persons must reside in a separate table, linked to the table of publications, with references to the groups and institutes they belong to; and that the names of the authors must be selected from a list when the publication entry is being created. (For reasons of uniformity, the same applies to the editors of books or conference proceedings, and to the reviewers or supervisors of doctor’s or diploma theses.) Obviously, it must be possible to add new names to the name table in the course of entering a publication.

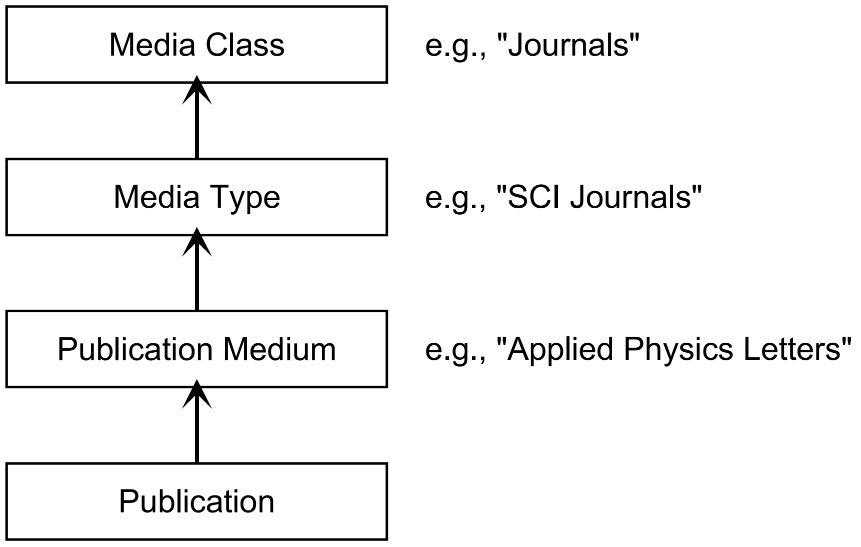

Fig. 1: Hierarchic organisation of publications in the Publication Database

“Weighing” publications should also be as easy as possible: It simply would not do to have information such as the SCI status of a publication or the impact factor of the journal in which it appeared entered separately for each publication. These are properties of the “publication medium” (e.g., the journal), which properly belong into a publication medium record (see Fig. 1). Similar to the names of authors, publication media have to be selected from a list, and only added to this list if they are not yet in the database. It should also be possible to tie together publication media with a comparable quality and regard them as belonging to one “media type” (which, in turn, determines their “weight” in an evaluation). For example, “journals listed in the SCI with an impact factor of greater than 1” may constitute a particular media type. Since journals and, e.g., conferences obviously cannot share media types, they constitute different “media classes”. The media classes recognised in the Publication Database are journals, publishing houses (for books and contributions to books or proceedings volumes), events (for oral or poster presentations at conferences or other scientific meetings), and patents. The publication media concept is not used for simple publications like diploma or doctor's theses or reports, which also should be represented in the Database.

The publication media concept greatly facilitates a sanity check of the data entered: Instead of looking at the classification in hundreds of publication entries, only the classification of the publication media needs checking. Particularly in the case of journals, the number of publication media grows only slowly after an initial phase, and it is easy to look up these newly added journals in the proper databases. (Of course, this process could be automated altogether.)

Publication references should be standardised but require a different structure for different types of publications. (In fact, a standardised reference format is one of the greatest benefits in creating publication lists from a database. Who ever had to create a decent looking publication list for a department report from lists supplied by various colleagues will probably agree.) It makes therefore sense to define "publication types": A publication type determines not only the format of the reference output; it also determines the media class to which the publication media offered for selection must belong.

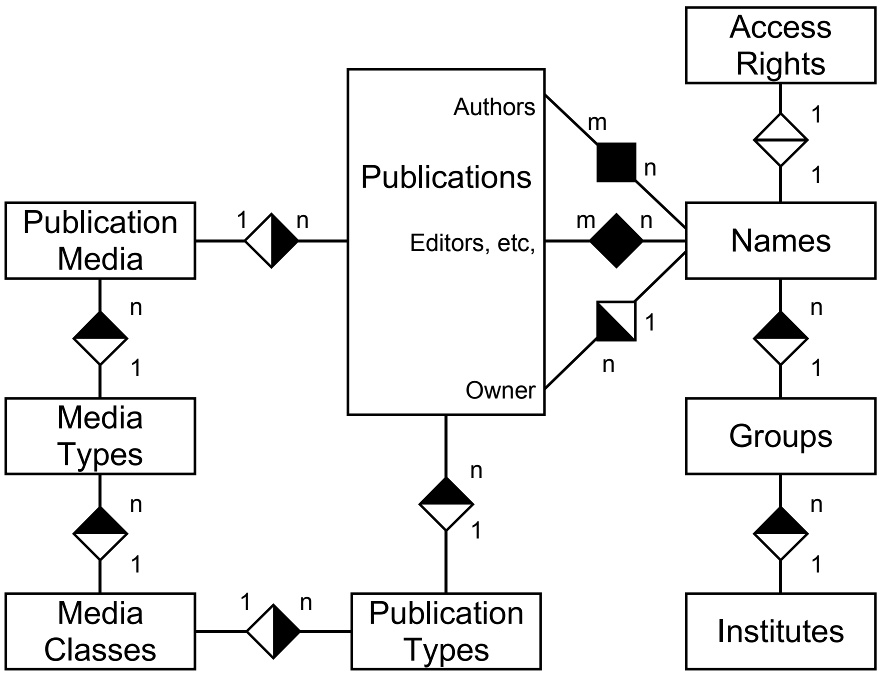

Fig. 2: Simplified ER diagram of the Publication Database

This structure results in the ER diagram shown in Fig. 2, which is a greatly simplified representation of the actual table structure of the Publication Database. Figure 2 does not show the tables that hold auxiliary information such as the formatting of the reference output, the grouping of publication types in publication lists, or the evaluation queries and results, and it also shows only one relation that determines the "owner" of a publication entry (i.e., the person that made the entry). All tables regular users can modify have similar fields that permit to determine who the last person to change the entry was.

The concept already introduced in the *Access* prototype to keep as much configuration information as possible in database tables proved to be exceedingly successful: No changes of the program code proper are necessary to introduce new publication types; this requires only adding records to the publication type and the formatting tables. In fact, the core table structure as shown in Fig. 2 has remained unchanged through the life of the Database; however, many new fields have been added to these tables.

The Implementation of the Publication Database

While the already mentioned shortcomings of *Access* dictated a different solution in any case, other boundary conditions favoured a Web solution over any other client-server concept:

- We were looking for a sustainable solution that should exceed the lifetime of common client software applications.
- In a university environment, there is a wide range of hardware and operating system platforms. This precludes conventional LAN-based clients.
- Using the Database as a knowledge base and providing external access to the publication information requires a Web interface anyway. This also applies to the "added value" of the Database as an on-demand generator of publication lists for the Web servers of the faculty's institutes.

In general, using conventional Web browsers as clients and the HTTP (or secure HTTP) protocol for transport makes the Database platform-independent and worldwide accessible. Since university members tend to use a variety of browsers, including some "exotic" species, browser-independent programming is mandatory.

For various (primarily financial, but also technical) reasons, we chose a LAMP structure for the database server, with client-based JavaScript for local pre-processing. There are three major access points to the Publication Database:

- An authenticated access for data entry and maintenance (the "administration module");
- An interactive public interface that allows searching for publications and/or creating tailored publication lists of persons, groups, or institutes; and
- A function that dynamically creates pages with publication lists in a custom design for inclusion on other websites.

In the two interactive interfaces, a variety of query functions permits restricting a search to entries meeting certain conditions. Both interactive interfaces provide a full-text search, which may comprise the entire entry including abstracts etc., or only certain fields of the entry. The administration module is in German only but permits to create publication lists in English; the other two interfaces are available in German and English. Although the administration module allows anonymous access, the recommended access for the outside world is the (much simpler) interactive interface.

A five-level access privilege scheme has been implemented in the administration module of the Database: Anonymous users have strictly read-only access. At the lowest authenticated level, users may create publication entries and edit their entries (where "their" means those entries they made themselves, plus all entries in which they appear in the list of authors). The third level extends the editing rights to all publication entries created by members of the group the user belongs to, or with authors belonging to this group. The fourth level analogously extends these rights to the user's institute. The highest level is the administrator who can edit any publication entry in the Database. A separate privilege attribute allows a user to change evaluation-specific parameters; this attribute is not technically but in fact tied to the administrator rights level. Since access to a publication also depends on the relation of the user to at least one of the authors, the table in which the access rights are stored is closely linked to the table that holds the names of authors and other persons shown in publication entries (see Fig 2).

Two different statistics functions provide evaluation data, (a) by the "official" algorithm, which is more or less a simple count of publications, however, with a very peculiar grouping, and (b) with a proprietary function that also accounts for the quality of the publication media and the sizes of the publications.

Since the "official" algorithms are not only rather convoluted but also likely to change unexpectedly, we chose a very flexible approach that permits an administrator to construct queries interactively based on publication and media types plus additional information from the publication entry. These queries are stored in a set of database tables for later use. Every user of the administration module may inspect the queries (which can be altered by an administrator only) and interactively run them. For the "official" algorithm, a publication counts for an institute if at least one of its authors belongs to the institute.

The proprietary function uses freely definable weights for each media type, which allows distinguishing between publication media with, e.g., different impact factors. In addition to the weight per publication, a weight per page can be specified (with a freely definable limit), and other facts enter as well via weight factors, e.g., whether a publication was invited or not. Depending on the configuration chosen, the points thus earned by a publication may be given to each group or institute to which at least one author belongs, they may be given in full to each author of the publication, or be split into equal parts between all authors. Since the latter approach tends to discriminate authors who work in groups, a simple algorithm depending on the number of authors can enhance the number of points before they are divided between the authors. While only administrators with the special privilege level may define the configuration, all users of the administration program can inspect it and run queries.

Additional functions of the Publication Database comprise a tool to create URLs for inclusion on other websites that request a certain selection of publication data from the Database, various database maintenance and integrity testing functions, and functions for extracting export files for the "official" evaluation procedure. While the URL generator is available to all users of the administration module, only administrators may access the latter functions.

The program structure chosen keeps most of the processing in the server-based PHP code. This facilitates software management and provides a secure and reliable processing environment. Most of the JavaScript code in the Publication Database is there to enhance the usability of the user interface. One example is presetting certain form elements if other elements were edited (e.g., to set a radio button to limit the database search to entries in a certain time range if the start or end times of the time range were changed). Other important features are a quick search through long lists of person or media names, or checking the completeness of an input form. Due to problems encountered with some browsers that had difficulties handling large amounts of data as JavaScript variables, some parts of the initially rather extensive client-side JavaScript data pre-processing code meanwhile either have been re-written or moved into the server-side PHP code. Likewise, all potentially security-related JavaScript functionality has been converted to server-side PHP. This applies, in particular, to the pre-processing of data that is to be stored in the MySQL database; having this done on the server side reliably protects the system from data intentionally designed to disturb or damage the database. Although we have been careful to avoid all but the most established JavaScript functions, there are still occasional problems on newly introduced browsers. (For example, *Opera* 7 chokes on JavaScript code about which no other browsers, including its own predecessors, complained.) This makes it advisable to convert JavaScript functions into PHP code wherever possible.

With one exception, the Publication Database intentionally does not distinguish between browsers: There is no approach to represent Greek characters that all browsers honour. Netscape 4 requires and Internet Explorer accepts that the corresponding ASCII character is output with the font set to “Symbol”, while the browsers with the Gecko engine (Netscape 6+ and Mozilla) exclusively require (and Internet Explorer accepts) the HTML entity representation. (Greek characters were necessary for the users in chemistry and mathematics.)

Unfortunately, the Web version of the Publication Database has been developed under PHP3, which precluded object-oriented programming and introduced some security hazards that could be resolved only recently. A consequence of the procedural programming technique used is a rather convoluted program code that is difficult to maintain.

The Publication Database was originally designed for use by one faculty only. When the university authorities chose to introduce it university-wide, we decided to implement one separate copy of the Database for each of the faculties and the large sub-groups of the Faculty of Science and Informatics. The resulting ten databases reside on the same physical server; they are individually accessed via the virtual Web server concept of Apache. Although the maintenance of ten separate databases requires more effort, compared to one database for the entire university, several reasons favoured the current solution:

- It does not make a difference for people entering publication data if they log into a university or a faculty Publication Database.
- Evaluation data are primarily gathered faculty-wise. Splitting the Database in the way chosen does not constitute a problem there.
- Faculties may want to use individual configurations of the Database. This is much easier to implement in separate copies of the Database (although we tried to prevent this so far to avoid incompatibilities between the databases).
- The lists of already registered authors and of the publication media with a suitable media class must be sent to the client browser each time the publication editing form is opened. These lists grow rapidly; in the EE Database, which holds the faculty’s publications from 1996 on, there are currently about 3,800 name and 2,100 media entries (for 6,700 publications). This amounts to page sizes of several hundred kilobytes; extending this concept to the entire university would increase this figure by about an order of magnitude, which is obviously unacceptable. Furthermore, finding suitable name or media entries in lists of that size is impractical.
- Currently, external visitors have to search in several databases where the publications they are looking for might appear. This drawback can easily be resolved, though, by introducing a portal that automatically searches all databases in turn.

Apart from a few configuration files (which define, e.g., the MySQL database to be used and the name of the faculty to be shown in the output pages) all copies of the Database use the same set of files. Unfortunately, the Linux environment does not permit to install the common tree of PHP, HTML, and image files only once and access it via symbolic links from each faculty's base directory where the configuration files reside; therefore, new software releases must be copied to ten faculty directories (which has been automated, though).

Experience Gathered with the Publication Database

As could be expected, people at the institutes initially met the Publication Database with (at least) suspicion, as an instrument designed to further increase their workload. We could alleviate their objections by pointing out the "added value" of on-line publication lists and queries, and by the promise that all publication-related evaluation data would come from the Database, without bothering them with these surveys in the future.

While the structure of the Database itself was quite well accepted by the EE faculty (with whose requirements in mind it had been designed), there were some objections by other faculties, where different approaches exist. In a rather odd coincidence, two different researchers claimed on the same day that the Database requested too many data fields, and that there were too few fields for bibliographic information, respectively. Due to the flexibility of the Database design (and of people working at a university), it was possible to provide solutions everybody could live with.

Some important extensions to the Publication Database were mandated by its university-wide introduction: Particularly users in chemistry and mathematics wanted Greek characters, superscripts, and subscripts in title (and abstract) texts. Greek characters are a problem because both PHP and MySQL only support eight-bit characters; the only feasible solution was to use a proprietary encoding scheme. (We are aware that proprietary solutions should be avoided if possible. However, both the HTML entity notation and the TeX notations for Greek characters were not suitable: the first, because browsers understanding Greek HTML entities convert them into their Unicode representation if an entry is opened again for editing, and the latter, because the backslash character used in the TeX notation simply does not make it through a form submission and the MySQL database.)

The already existing electronic collections of publications at other institutes constituted a particular problem: Around the time when we introduced the prototype Database at the EE faculty (1999), several institutes and even faculties had started similar activities. Obviously, these institutes wanted to continue using their data collections, or at least to import their data into the Publication Database. These data collections ranged from simple *Word* documents over BibTeX files to more or less well-structured databases. Some of the requests were surrealistic indeed: the Informatics people (who should have known better) demanded that all existing publication collections regardless of format and structure must be imported fully automatically. The university hired a programmer to create an import tool for publication collections; in the end, it turned out that – except for well-structured database or BibTeX sources – the effort for post-processing the import data to make them suitable for a fully structured database came close to newly typing in these entries. In particular, most problems we encountered originated from non-uniform publication entry formats in many of the collections; in some cases, each reference entry differed from its neighbours in punctuation, order of first and last names of the authors, and available data items.

Apart from the proper fault-free operation of the Publication Database, the usability of its user interface has been the most important design issue, beginning with the *Access* prototype. Data entry forms should be as clear and self-explanatory as possible, and the sequence of operation steps transparent. The design of the *Access* prototype and the experience gathered with it, in turn, influenced the design of the Web version. Some enhancements became necessary only after a prolonged operation of the Database, particularly due to the growing sizes of name and publication media lists. The problems with these lists were addressed in two ways: either by reducing the amount of data in the list where possible (e.g., by showing only the names of people belonging to the currently selected institute or group), or by providing a "quick search" for entries in these lists. Some seemingly insignificant features greatly facilitate work for the users, e.g., the possibility to sort entries by age (with the latest on top of the selection list), or to limit searches to entries that still require some action.

Some problems originated from the wide variety of hard- and software platforms and browser configurations used university-wide. Although we had spent much effort in making the Publication Database as platform and browser-independent as possible, we could not test it with every conceivable browser configuration and network structure. A relatively high degree of browser-independence is possible by using very basic HTML and JavaScript programming, avoiding sophisticated style attributes and JavaScript functions. Although the resulting pages may not look as uniformly on those browsers that support high-level style attributes, they at least work on all reasonable browsers. While the administration module, which requires client-side JavaScript, needs at least *Netscape* 4 or *Internet Explorer* 4, the public interface can also operate without JavaScript (although it has a smoother user interface on JavaScript-enabled browsers); in fact, you can even use *lynx* to work with the public interface!

The combination of peculiarities of a particular browser with its usage by particular persons created some odd effects that were very difficult to trace and reproduce. It turned out that it were generally the same users that caused fatal errors in the Database, while other users with the same browsers remained inconspicuous. For example, *Netscape* 4 allows submitting a form that is not yet completely loaded (which runs havoc if essential "on submit" JavaScript code is not yet there), while *Internet Explorer* permits multiple submits of the same form. This is usually not an issue, but made itself felt with the huge pages for publication entry editing, particularly if there was a slow network link, and with very impatient users. (Fortunately, we could identify and resolve this problem before the Publication Database went into university-wide operation.) Another issue that may severely reduce the usability of a Web application is a different interpretation of pressing the "Enter" key, especially if there are several "Submit" buttons in a form (which is a common situation in the Publication Database if different functions may be invoked with the same data). While some browsers (e.g., *Netscape* 4) ignore the "Enter" key altogether in this case, others (e.g., *Mozilla*) interpret it as a click on the first "Submit" button in the form that has the focus. Since it is likely that the first submit button was *not* the button the user wanted to operate (at least at the time she inadvertently pressed the "Enter" key at the end of a text field), we have added dummy "Submit" buttons on all frequently used forms that are likely to get the focus because they contain text fields. These dummy "Submit" buttons simply re-draw the form with its current contents, which is probably the most benign reaction in this context.

We found that a thorough checking of data entered by the users was essential: Regardless of academic degree and scientific orientation, users tend to make all conceivable mistakes. Several enhancements of the data checking mechanisms became necessary after the Database had been introduced university-wide; even if it comes to making mistakes, the creativity potential of academics should not be underestimated.

The Future of the Publication Database

Although the Publication Database by now has reached a very high level of functionality and completeness, there are several reasons that make a complete redesign desirable:

- The Database has been designed as a stand-alone application. Major modifications would be necessary to embed it into an integrated university information system, which should happen in the near future.
- With its procedural PHP3-based code and with the wealth of functionality meanwhile packed into it, the Publication Database is extremely complex and difficult to maintain. The university IT administration who should conceptually be in charge of the Database to date was not able to take over the Database (and even flatly refused to do so). Therefore, the author is still faced with its maintenance, which is not a desirable situation at all.

A re-write of the Database must, of course, duplicate its current functionality and add some more, among others:

- Integration with other university-internal and external databases;
- Smart reduction of the amount of data to be sent to the Web clients; and
- A few features in the handling of publications and the creation of publication lists that appear desirable but cannot be implemented in the current structure.

Conclusions

The experience made with the Publication Database indicates that a few simple but important design rules greatly improve the usability and adaptability of a Web application, even if it has to operate in an environment it was not originally designed for:

- Use generic code on the server, and put all configuration information into external data, either into a database table, or into configuration files;
- Send minimalist HTML and JavaScript code to the client browsers;
- Support a reasonable range of standard browsers; and
- Do not assume that all of your users read pop-up messages, let alone help files and software documentation. Provide syntactic and, if possible, heuristic checking on all data entered by the users!