6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Building a Web-based Health Promotion Database

Ádám Rutkovszky

University of Debrecen, Faculty of Economics
Department of Economics

Abstract

A survey of health promotion programs in Hungary was conducted by the School of Public Health at the University of Debrecen's Medical and Health Science Centre. A questionnaire was sent out to institutions and organizations. A data model for storing information on the completed questionnaires was developed, and a database was built up. A web-based application was developed for both querying and administering the above-mentioned programs.

This paper investigates some crucial questions that may arise during the development of such applications. We describe the problems related to user authentication in web-based solutions, which is essential when using limited access or web-based administration. Data validation techniques that can be applied in the case of form-based user input are also examined. General application-level web security problems including form modification and SQL attacks are described. To achieve greater publicity for a web page or web site it is important to register it with search engines. The submission of information contained in databases to search engines raises some interesting questions, and is also discussed herein.

An example is given for the implementation of a possible solution to each of these problems, also describing the hardware and software environment of the application created.

1. Introduction

Since the beginning of the 90's the World Wide Web has acquired a great popularity. The web has evolved from a handful of sites containing mainly static pages to millions of information repositories fed by databases. The web has become a standard interface for sharing information across geographical boundaries.

Organisations publish their knowledge on the web using dynamic pages, the content of which is stored in databases. Several scripting languages have been developed for accessing data stored in databases and displaying them on web pages.

One of the most popular subsets of these languages is the group of open source server-side scripting languages. According to a Netcraft survey [1] from January 2004 PHP is used at more than 14,000 domains. A report published on 1 January 2004 by SecuritySpace [2] showed that PHP is the most popular of the Apache modules, with a penetration rate of nearly 54%. This makes PHP the most popular open source server-side scripting language on the Apache platform. The trio of Apache+PHP+MySQL can be regarded as a common technology for "low budget" web application projects.

The development of applications displaying content stored in databases using server-side scripting presents us with some common problems. This paper examines these problems and explores the possible solutions. Implementation is described using an operating web-based database application called Health Promotion Database as an example.

The rest of the paper is organised as follows: section 2 gives an overview of the Health Promotion Database. In section 3 we discuss user authentication solutions. Section 4 describes problems of form handling. In section 5 we explore difficulties of submitting database content to search engines, and finally, section 6 gives a summary of the main conclusions and future tasks.

2. The Health Promotion Database

A research on health promotion programs in Hungary was conducted by The School of Public Health at University of Debrecen, Medical and Health Science Centre. A questionnaire was sent out to institutions and organisations. The completed and returned questionnaires were categorised according to the following criteria:

- City
- County
- Complexity
- Data-provider
- Coordinator
- Site
- Education of the target group
- Employment of the target group
- Other characteristics of the target group
- Health field
- Total cost

Categories city, county, complexity, location, education of target group, employment of target group, health field can have multiple values.

A data model for storing the information from completed questionnaires was developed and a database created. The questionnaires were converted to pdf format. The software platform was provided by The School of Public Health and had the following characteristics:

- Operating system: Microsoft Windows NT
- Web-server: Microsoft IIS
- Database management system: Microsoft SQL Server
- Scripting language: PHP 4

The web application for publishing information on the returned questionnaires basically consists of two modules: a query module and an administration module.

The query module [Figure 1.] can be regarded as the public part of the application since it is accessible without any restrictions. This module provides functionality for searching the database. The administration module [Figure 2] has restricted access and provides functionality for managing programs and properties.

Users visiting the site can search the database by filling in a search form (search_form.php) and sending it to a script (search.php), which then creates the appropriate SQL queries and sends them to the database. The database runs the queries and sends the results back to the script, which finally generates the HTML output.

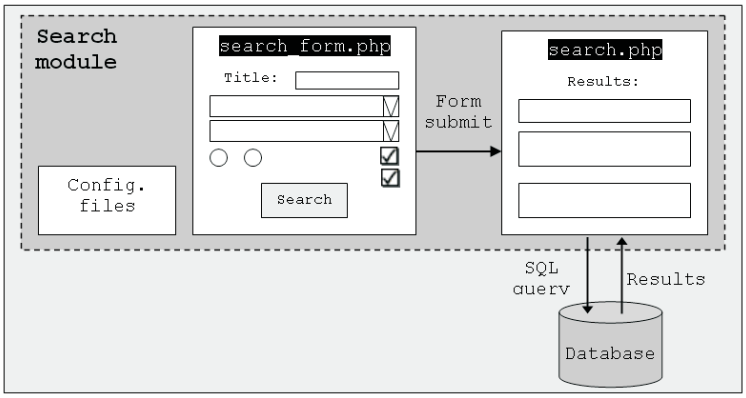

Figure 1.

Application administrators can upload programs using the web interface of the administration module. Pages belonging to this module can be accessed after successful authentication. After successfully logging in, the administrator can select the following options from the main menu:

- Upload new health promotion program,
- Delete health promotion program stored in the database,
- Add new value to a category,
- log-out

The number of health programs currently stored in the database is also displayed on the main menu.

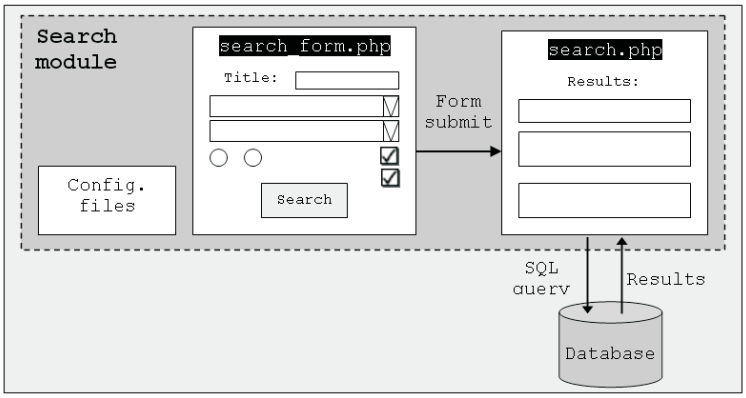

Figure 2.

3. User authentication

Authenticating users is a basic requirement when using web-based administration in a web application.

A possible solution for protecting pages of the application is to use the tools provided by the web-server or the operating system. Limiting access using.htaccess files on Apache web-servers is a possible means of forcing users to identify themselves. The drawback of this method is that the valid username/password pair is not stored in the database. The fact that the password and the username given by the user are sent plain (i.e. without encryption) is another disadvantage of this solution.

Basic HTTP authentication uses a challenge/response scheme to authenticate users attempting to access a password-protected page. The challenge process begins when the user requests a file from a Web server. If the file is within a protected area, the server responds by sending out a 401 (unauthorized user) string in the header of the response. The browser detects that response and displays the username/password dialog box. The user enters a username and password in the dialog box, then clicks OK to send the information back to the server for authentication.

If the username and password pair is valid, the protected file will be displayed to the user. The access rights thus gained will be valid for as long as the user remains within the protected area. However, if the username and password typed into the dialog box cannot be authenticated, the dialog box will be displayed again, prompting the user to try again. This cycle will be repeated until the proper username/password combination is entered or the user gives up and slinks away. Validation of the username and password pair can be performed against values hard-coded in the string or values stored in the database. Basic HTTP authentication seems to be a better solution than using web-server provided functionality but also has some drawbacks, since it can't be used when using CGI version of PHP. In our case we have the CGI version so we had to find a different solution.

A database-driven authentication method has been developed for the Health Promotion Database, based on HTML forms for inputting the username and password, and PHP for validating the given pairs and cookies to store login information. Every page within the administration module contains code that checks whether the user is logged in or not. If not, then the browser is redirected to the page containing the login form. As soon as a valid username/password pair is given the browser receives a cookie with information indicating that the user is logged, and including a timestamp. This timestamp guarantees the need for re-authentication after 10 minutes of inactivity. A logout link is placed on every page of the administrative module. When this link is clicked the logout script sets an invalid timestamp in the cookie and the browser window is forced to close.

4. Form-handling

The most common way of inputting user data to web applications is form-based. As in our Health Promotion Database, users fill in a form, which is then submitted – at the click of a button – to a processing script. In most cases data validation is carried out at the client-side after clicking the submit button, but before submitting form-data to the processing script. This algorithm seems to guarantee that all data posted to the database is valid. It should be borne in mind that form-data could be encoded in the URL of the processing script (in case of a GET method page), and therefore server-side validation is also necessary to ensure that the data used for creating queries satisfies the set conditions. Constructing the query naïvely from data posted by the client leads to a vulnerability whereby the user can execute arbitrary SQL against the back-end database. The attack is best illustrated with a simple example:

Consider an Employee Directory Website written in PHP, which prompts a user to enter the surname of an employee to search for by means of a form-box called searchName. On the server-side this search string is used to build an SQL query. This may involve code such as:

```
$query = ‘‘SELECT firstname, tel, fax, email FROM person WHERE
lastname=’$searchName’";
```

However, if the user enters the following text into the `searchName` form box:

```
‘; SELECT password, tel, fax, email FROM person WHERE
lastname=’Rutkovszky
```

then the value of variable, $query will become:

```
SELECT firstname, tel, fax, email FROM person WHERE lastname=’’;
SELECT password, tel, fax, email FROM person WHERE
lastname=’Rutkovszky’;
```

When executed on some SQL databases, this will result in Rutkovszky's password being returned instead of his last name. (Even if only a hash of the password is leaked, a forward-search attack against a standard dictionary stands a reasonable chance of recovering the actual password.)

5. Submitting database content to search engines

Achieving greater publicity for a web site or web application requires submitting the URL of the starting page to search engines. After submitting a URL a spider tries to crawl the given sites. Using only static pages the spider will be able to crawl the whole site following the links. Problems will begin when pages are generated dynamically from the content of a database. Such pages are accessible for users by following pages containing questions marks (?) or using forms to search the database. When following links to dynamic pages it is quite possible for a spider to get into a loop from which it cannot escape, therefore spiders such as Google's and HotBot's will follow such a URL only to the first level. On the other hand, spiders can't fill in forms which makes content that only accessible through a form invisible to them.

However, although hidden web crawlers do exist, they are not used by the widely known search engines. This means that webmasters or authors of sites have to make their pages visible to spiders. A webmaster may choose to optimise his or her site so that all links to dynamic pages appear as 'spider-friendly' static links. There are various URL rewriting tools that assist in doing this. Many of these tools allow the '?' in the URL to be replaced by a different character like '/'. The drawbacks of this method are cited in [3].

Creating a table of contents (TOC) – a page containing links to the content stored in the database – can be regarded as a possible solution to the problem of invisible pages. In order for the TOC to be effective, it must be found by the search engine spiders. If the default home page contains a link to the TOC page, a spider will navigate to the TOC and then be able to find all linked content. A site

map page is also a good place from which to link the TOC. The TOC itself can either be created by hand which requires significant manual effort or be outputted dynamically by a script.

In the case of the Health Promotion Database, the solution of generating a TOC page linked from the home page has been chosen. The script for performing TOC generation has yet to be written. It will run after successfully uploading a new health promotion program with the help of the administration module.

6. Conclusion and Future Work

In this paper we have presented a web-based Health Promotion Database. We have also demonstrated the general problems of developing web-based database applications in server-side scripting languages. Problems of user authentication and form handling have been discussed, and the difficulties involved in submitting database content to search engines have been shown. We have put forward the idea of creating a table of contents as a viable solution to these problems. Future tasks includes implementation of the TOC.

References

[1] Netcraft: em Web server survey.
`http://news.netcraft.com`

[2] SecuritySpace: *Apache module report.*
`http://www.securityspace.com`

[3] YourAmigo: *Spider Linker White Paper* (YA4.010 Rev. 1.05).
`http://www.YourAmigo.com`