

HCI 590 / CSC 459 Database Management Systems  
Independent Study Project

**WebStats – HTML Web-Access Statistics Engine**

**Final Project Report**

Bastian Tenbergen  
Human-Computer Interaction MA Program

State University of New York at Oswego

**Abstract.**

Goal of this Independent Study was to acquire and apply real world knowledge in the field of Database Management Systems. It was meant to serve as a platform to facilitate the understanding and autonomous learning of the subject of Database Management Systems. In order do so, a software has been developed that incorporates as many aspects of DBMS as possible. This software – WebStats – can be used to retrieve data in order to conduct Data Mining.

**Table of Contents**

1	Introduction and Project Requirements .....	Page 03
1.1	Project Goal .....	Page 03
1.2	Project Requirements .....	Page 03
1.3	Project Software .....	Page 03
2	Implementation Details .....	Page 04
2.1	Model – The Database Design .....	Page 04
2.2	View – Configuration and Deployment .....	Page 05
2.3	Controller – WebStats' Engine .....	Page 06
2.4	Web-Accessible Components .....	Page 07
3	Implementation Review .....	Page 08
3.1	Project Strengths .....	Page 08
3.2	Potential Pitfalls .....	Page 08
4	Conclusion .....	Page 09
5	References .....	Page 11
6	Appendix A – Normalization .....	Page 12
7	Appendix B – WebStats User Manual .....	Page 14

In order to save a tree, neither Javadoc Documentation nor Source Codes of WebStats have been attached to this document. They are available for free on <http://moxie.cs.oswego.edu/~tenberge/hci590/>

## **1. Introduction and Project Requirements.**

This section discusses the initial specification of the project. Specifically, it lists the goals and requirements that this project aimed at fulfilling.

### 1.1 Project Goal

Goal of this independent study project was to serve as a platform to facilitate understanding of the principles, theories and procedures regarding Database Management Systems. A project software was developed that incorporates as many aspects of DBMS as possible, thereby deepening the understanding of the matter and allowing to acquire practical knowledge. This is an incremental process and lasted the duration of one semester.

### 1.2 Project Requirements

This project comprised, the understanding of the following aspects of Database Management Systems:

#### **Database Design.**

This project deals with the design and planing of a suitable database schema. Special attention will be paid to the aspects of maintainability, performance and extendability.

#### **Database Interaction.**

A client software and web front-end was developed that allows for database interaction. The software is used to manipulate and query the database.

### 1.3 Project Software

The result of this project is a website analysis tool that uses a Database Management System to retrieve and store data about the website traffic and the traffic statistics over time. Despite the fact that there exist already ample traffic analysis tools, this tool will differ from the broad mass of already existing products in such that it allows to monitor the traffic of a website and monitor the changes in the website content and thereby give users an understanding of how changes in the content of the website influence the traffic to the site. The Project Software therefore covers the following features:

### **Automatic Content Tracking.**

The software automatically tracks the content of a certain website and add it to the database. A user-centered approach for this software component ensures optimal user friendliness, but Usability Testing was not be conducted, as it was beyond the scope of this project.

### **Automatic Representation of Data.**

A web front-end of the database was implemented and show the collected data. This feature comprises a number of customizable PHP files that can be used to query that data of in the database and allow for manual inspection using a graphic representation (i.e. HTML tables) rather than command-line based queries.

## **2. Implementation Details**

This section discusses details of the implementation of the project software “WebStats”. The final version of the product is explained in terms of functionality and purpose. WebStats has been developed following the Model-View-Controller paradigm, even though it is not a traditional user-interface-equipped software. Since WebStats is mostly a collection of background processes, the interface to the user consists mostly of the configuration files.

### 2.1. Model – The Database Design

The core aspect of this application is the database. The Database Management System of choice is MySQL, version 5, as it implements the SQL standard, is free of charge, and is natively supported in a multitude of high-level programming languages and scripting languages.

The database at hand has been designed according to the Entity-Relationship model (see Appendix A for the E-R diagram and Normalization). The two core entities are server data and client data, representing files stored on the server with their associated attributes, and data regarding connections, which view a certain file on the server. Information such as file size, date of last modification (implemented using a multivalued attribute to keep track of the modification history of the files), and the path is stored in a relation for every file on the server. For each client that connects to the

website, which is being tracked using this software, information are being extracted and sent to the server. Among others, these information include Operating System type, System language, current time, location, file that is currently used and alike. For a full reference, refer to Appendix A. Each set of client information is stored in a relation. Using a relationship set, it is recorded which client is currently viewing what file on the server. Since every client-connection can view one and only one file on the server, it follows that the client data entity set participates totally in the underlying “view” relationship and that the server data entity set is in a one-to-one-relationship with “view”.

This implementation paradigm results in four central relations in the database: one relation for the entity set of the server data, one relation for the entity set of client data, one relation for the relationship set of server data and client data and one relation implementing the multivalued attribute of modification dates for the server data. Additional temporary relations might be present while WebStats' background processes are running and are not documented outside of the source code. Details regarding implementation and normalization can be found in Appendix A.

## 2.2 View – Configuration and Deployment

Since it is traditionally difficult to configure and deploy daemon-based software, an effort was made to centralize the configuration procedure to one file. As section 2.3 states, WebStats consists of not only the database and some PHP files that must be configured to comply with a database, it also consists of two independent background processes that manipulate the database and a HTML/JavaScript combination that enables HTML documents to be tracked using WebStats. Due to the multitude of parts and technologies that WebStats consists of, configuration is somewhat complex. However, an effort was made to make the configuration as simple as possible. For a detailed description of how to configure and deploy WebStats, please refer to Appendix B.

Aside from all web-related configuration (i.e. configuration of PHP files and HTML documents), the core part for configuration is a configuration file. In order to make configuration as simple as possible, WebStats automatically parses this configuration file and sends the necessary information to every part of the program. Without the centralized configuration file, it would have been necessary to recompile WebStats every time it gets deployed to a different server, since a multitude of classes use different configuration options (in terms of which database relation to use, where to store

temporary data, etc). The paradigm of having a centralized configuration file also relieves the user from the burden of manually starting all independent background processes separately. The same class responsible for parsing the configuration file takes care of launching the needed parts of WebStats at the correct time when it is appropriate to be started. This makes executing WebStats highly simplistic once WebStats has been correctly configured.

Although the notion of configuration is not traditionally understood under “View” part of the MVC paradigm, it can be understood as such in the context of a software that is merely running as a daemon, as it does not expose a GUI to the user and the only interaction with the user is through configuration.

### 2.3 Controller – WebStats' Engine

The core of WebStats is a collection of Java classes that carry out the basic functionality. It essentially consists of three independent parts that work in unity.

The first part is a simple Launcher. It is invoked when WebStats is being executed in the Java Virtual Machine. It checks the configuration file for consistency and parses it, keeping a reference to all configuration variables. The Launcher exposes these configuration variables to all parts of WebStats that need it. After parsing is complete, the Launcher takes care of launching the other two parts in separate threads. Specifically, it takes care of executing the FileCrawler every 10 seconds and running the HttpServer in a ThreadPool to assign incoming connections to separate threads.

The second part of WebStats is the FileCrawler. This device has to be run before any connections to the HttpServer can be made. The FileCrawler takes care of recursively scanning the directory with the web content (default: “www\_data”) for new and old files. It collects information on the files, such as filename, path, and file size and stores these information in the database. The files the FileCrawler encounters are stored in a folder (default: “server\_data”) so that a constant reference is kept for the file (it is added as a blob to the database). This is necessary to enable the system to keep track of changes in the file – it creates a new copy in the server data folder whenever the current timestamp of the file differs from the timestamp stored in the database. Also, the FileCrawler checks all the files it has found in the current iteration against the files it has previously found in order to determine, whether or not a file that has been previously found has been deleted or not. If a file has

been deleted, it should therefore be in the database, but should not have been encountered in the current iteration. If this is the case, the FileCrawler requests the Database Management System to set a deleted flag in the server data relation for the specific file that has been deleted to true.

The third part of WebStats is the HttpServer. Essentially, it implements a simple CGI-like server side action server. It is to note, however, that the HttpServer is **not** an CGI script. It essentially a simple, low level HTTP server that takes incoming requests, parses the argument URL to retrieve information sent to it. The information that are passed to this framework are details retrieved from the client browser of the user that is currently viewing a HTML file that is prepared to work with WebStats. Since the information are passed to the HttpServer in form of a URL, a server side action has been implemented that parses the information URL string and stores it in a database. The HttpServer also tries to correlate the information with server file data already stored in the database (i.e. those that have been inserted by the FileCrawler) in order to look up which file is currently being viewed by the client. **Note:** A framework has been developed for this HttpServer to allow future customization so that the HttpServer can perform other tasks than the ones described above.

#### 2.4 Web-Accessible Components

The last part of WebStats are all web-accessible components. These consist of a set of PHP templates and a JavaScript script along with an HTML template.

The JavaScript script “webstats.js” must be copied to the content that shall be tracked using WebStats. Please refer to Appendix B for deployment details. Also, every HTML file that needs to be tracked must be modified according to the description in Appendix B to be able to execute the script. The script itself is responsible for retrieving information from the user's browser, and sending it to the HttpServer. As described above, the HttpServer takes care of parsing the information string and entering the data into the database.

The PHP templates allow users of WebStats to view the database content without the need of command line access to the MySQL server. Given a user name and password that must be added to the PHP files, these files will show the content of the different tables to be displayed conveniently in a browser. All files can be customized. Since a variety of queries is possible to allow for different analysis and correlations along with other Data Mining tasks, only a few example PHP files have been included.

Given some fundamental MySQL knowledge, one can easily infer other queries to display different combinations of the data in the database.

### **3. Implementation Review**

This section serves as a review of the implementation of WebStats. Positive aspects and strengths of WebStats are discussed and potential pitfalls are pointed out before a conclusion on the Independent Study is drawn.

#### 3.1 Project Strengths

Despite the rather complex structure, WebStats has the capability to be an excellent tool for tracking statistics of web content of a website. Since a thorough set of data is collected from clients and server, and the code base is designed so that it can be scaled up easily, it can be conceived that WebStats can be used in a professional manner. The goal was to provide a product with a feature that competing web statistic analysis engines like Google Analytics are lacking, namely to provide an insight in how the change of the web content impacts on the visitor statistics. Since WebStats keeps track of previous versions of individual HTML files, it can be seen which file version attracted the most users per time period. Along with the fact that WebStats can easily be modified to comply with different data (that are collected from the server files and client browsers) and can easily be scaled up to track more than one website, WebStats presents itself as a robust and light-weight way to track web content without the need of extra hardware, software or administrative access on the host machine.

#### 3.2 Potential Pitfalls

WebStats is a very robust software. There are no known issues, bugs or problems, as an effort was made to deliver a software that is as close to production quality as possible. However, as with every project, a few compromises had to be made during the development.

The biggest issue is the problem of configuration and deployment of WebStats. Since WebStats consists of many different parts that work independently to provide the wide array of functionality that this software offers, configuration is a key issue. Since from a usability standpoint, configuration and use of the program are essential and must be optimized by lowering configuration and deploying

complexity. At the same time, however, it must be possible to deploy WebStats to any kind of web content and any kind of host system – a trade-off. The compromise that was made to approximate an ideal solution is the notion of a configuration file. Along with the Manual (Appendix B), users should be able to deploy the application conveniently.

Another problem is that clients cannot be uniquely identified. Although it was a requirement of the initial specification of WebStats, implementing this feature proved difficult. Despite the fact that the JavaScript component is able to retrieve IP-Address and host name information from the client (which was the initial solution to provide this feature), this JavaScript-based approach is not robust enough. Different browsers implement this information differently. To make this issue even worse, different versions of the same browser do not handle this feature consistently. The result is that IP-Address and host name information cannot be reliably retrieved, resulting in the default value (“127.0.0.1” for IP-Address and “localhost” for host name) being transmitted to the HttpServer in many cases, when the script was unable to determine the IP-Address and the host name of the client computer. A solution would be to place a cookie on each client machine. This can aid in identifying users uniquely. Since this was an Independent Study in Database Management Systems and not Web Design or Networking, this feature has been moved to future implementations.

#### **4. Conclusion**

This Independent Study can be concluded with one word: Success. During the implementation of a database-centric application, I was able to learn a lot of concepts, methodologies and paradigms of Database Management Systems. I was able to deepen my knowledge and understanding web services, concurrency, database transactions and security, database design and implementation. More importantly, I was able to acquire substantial knowledge of how to implement complex databases and evaluate the pros and contras of different implementation techniques. In addition, this project made me understand how to interface a database with various programming languages, such as Java and PHP. Furthermore, this Independent Study was an ideal opportunity to deepen my knowledge of previous courses I have taken – my knowledge in Concurrent Programming and Web Services was essential in the success of the software and the Independent Study.

WebStats is a robust product of high quality. Although I do not intend to continue the development of

WebStats, I will continue to use it to record statistics of my web appearances, such as the SketchUML Homepage: <http://sketchuml.tenbergen.org> . The goal of this project was to develop a product that will hopefully be used by a wide set of users and maybe even continued to be developed. WebStats is a product one can be proud of and one of the more impressive projects I have worked on, despite the few flaws it has.

Of course, my work with Database Management System is not over. Since database technology is essential in Software Engineering, Web Design and Computer Science, I believe that I will continue to work with these technologies for a long time. I am glad that this Independent Study was so successful as it helped me to gain significant insight into this technology, as I was completely unaware of the “nuts and bolts” of Database Management Systems prior to this Independent Study.

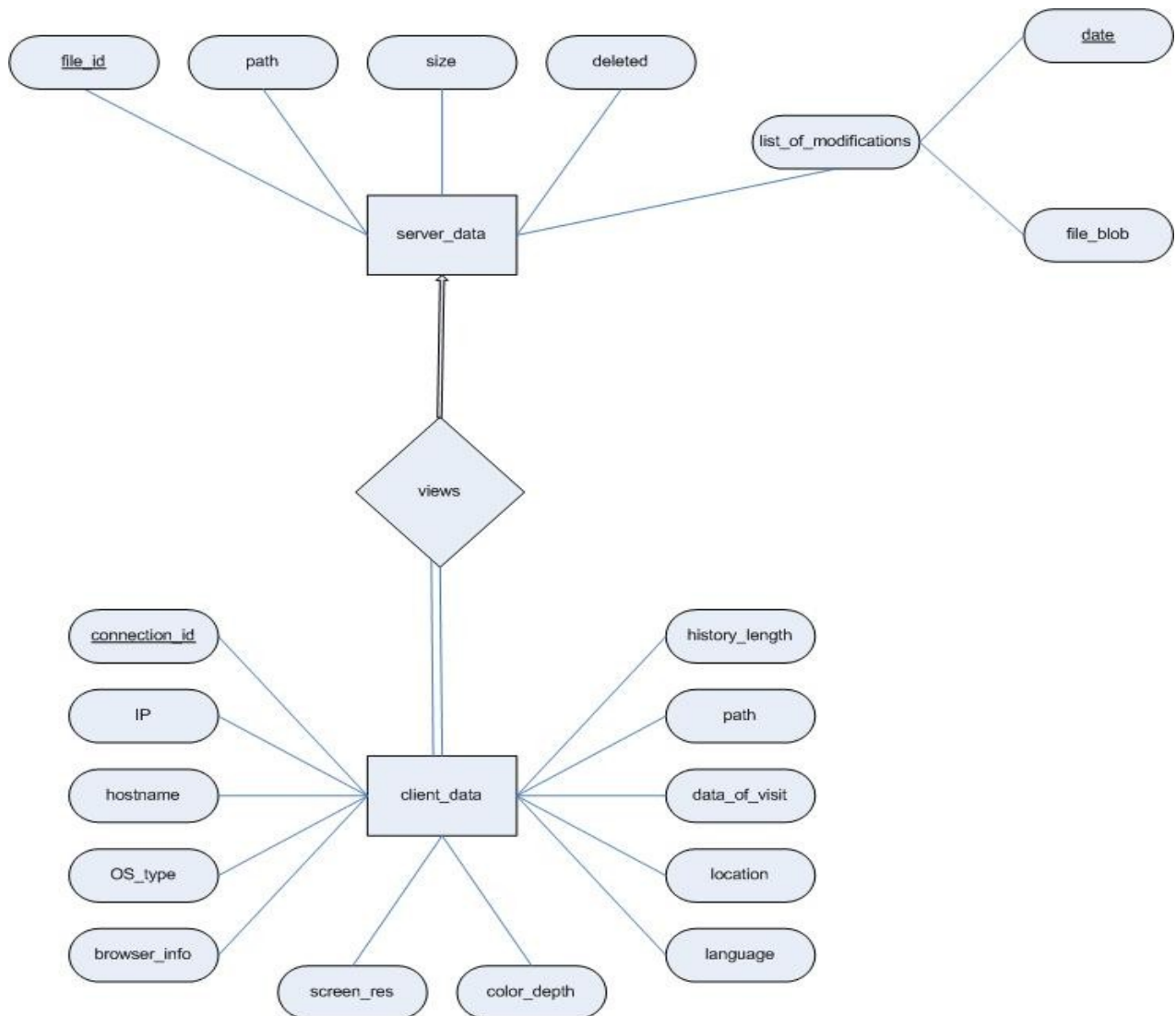
## 5. References

- Berry, M. J. A., Linoff, G. S. (2000). *Mastering Data Mining*. John Wiley & Sons, Inc.
- Goetz, B. (2006). *Java Concurrency In Practive*. Addison-Wesley Pearson Education.
- Graham, S., Davis, D., Simeonov, S., Daniels, G., Brittenham, P., Nakamura, Y., Fremantle, P., König, D., Zentner, C. (2005). *Building Web Services with Java, 2<sup>nd</sup> Edition*. Sams Pubilshing.
- Niederst, J. (1999). *Web Design in a Nutshell, 1<sup>st</sup> Edition*. O'Reilly.
- Lea, D. (1999). *Concurrent Programming in Java, 2<sup>nd</sup> Edition*. Addison-Wesley.
- Silberschatz, A., Korth, H. F., Sudarshan, S. (2006). *Database System Concepts, 5<sup>th</sup> Edition*. McGraw-Hill Higher Education.

## 6. Appendix A – Normalization

This document displays the database design and discusses the notion of normalization. The aim is to point out in which – if any – normal form the database has been designed in to ensure an optimal design reducing redundancy and maximizing performance. In the following, an Entity-Relationship diagram of the database can be found along with a formal definition of the relations. For each relation schema, a list of normal form constraints (given a desired normal form) that are satisfied is shown.

### Entity-Relationship Diagram.



**Relation Schemas.**

In the following, please find the relation schemas in this database.

server\_data = (file\_id, path, size, deleted)  
 list\_of\_modifications = (server\_data.file\_id, date, file\_blob)  
 client\_data = (connection\_id, IP, hostname, OS\_type, browser\_info, screen\_res,  
 color\_depth, language, location, data\_of\_visit, path, history\_length)  
 views = (server\_data.file\_id, client\_data.connection\_id)

**Normalization to Boyce-Codd Normal Form.**

From the relation schemas, it follows that given an antecedent, a set of precedents can be inferred. This section lists all possible conclusions that can be drawn from the antecedents and which constraint of Boyce-Codd Normal Form are satisfied. BCNF is the desired normal form for this database.

<u>Antecedent → precedent</u>	<u>BCNF Constraints satisfied</u>
file_id → path, size, deleted	√ Antecedent is primary key √ Conclusion is not trivial
file_id, date → file_blob	√ Antecedent is primary key √ Conclusion is not trivial
connection_id → IP, hostname, OS_type, browser_info, screen_res, color_depth, language, location, data_of_visit, path, history_length	√ Antecedent is primary key √ Conclusion is not trivial
file_id, connection_id → path, size, deleted, IP, hostname, OS_type, browser_info, screen_res, color_depth, language, location, data_of_visit, history_length	√ Antecedent is primary key √ Conclusion is not trivial

All antecedents are primary keys and no precedent is trivial. It follows, that this database is in Boyce-Codd Normal Form.

## **7. Appendix B – WebStats User Manual**

On the following pages, please find the official User Manual that is shipped with the WebStats binaries, source files, and project packages.

## WebStats HTML Web-Access Statistic Engine v1.0

# Manual

### Table of Contents

1. Introduction
2. Prerequisites
3. Configuration
  - 3.1. WebStats.conf
  - 3.2. webstats.js
  - 3.3. PHP files
4. Deploying
  - 4.1. Installing MySQL Database
  - 4.2. Installing WebStats-1.0-bin.jar
  - 4.3. Preparing HTML Files
  - 4.4. Installing PHP Files
5. Executing WebStats

### 1. Introduction

WebStats is a utility that parses the content of a web folder and records visitor statistics. It is essentially a three tier software – a simplistic low-level HTTP CGI-like server to handle client connections, a recursive file crawler and a MySQL database. The HTTP server and the file crawler run independently from one another; they only interact through the database. WebStats is entirely written in Java and is therefore absolutely platform independent. A number of sample PHP files are included that allow the user to view the recorded data. They are easily expendable to mine for different data in the database, depending on the web statistics recorded by WebStats. Any type of HTML document can be tracked using this software as long as the necessary modifications are made to the content about which statistics shall be recorded.

This document describes the basic configuration of WebStats and how to deploy it. An effort was made to keep WebStats as simple and easy to use as possible and to run with the bare minimum on requirements.

### 2. Prerequisites

In the following, please find a listing of the software that is needed by WebStats. An effort was made to keep this to the bare minimum. It is assumed that all these softwares are already installed and configured properly, as explaining these steps is beyond the scope of this document.

Required software:

- some web server (like Apache's httpd)
- some PHP extension for the web server
- Java Runtime Environment v1.6 or higher
- MySQL v5 or higher with a user account that has rights to insert, delete, alter and truncate tables

You will also need a user account on the host machine. Furthermore, the web server of the host machine must point to a directory that this user account has access to to be able to upload and track your own web content.

Things you do not need to run this software:

- root / administrator access on the host machine
- Jakarta / Tomcat or any other server capable of handling CGI or dynamic content
- MySQL Connector/J – this is already included in the WebStats distribution

### 3. Configuration

This section discusses how to configure WebStats and make all the necessary arrangements to deploy the application to your webserver. These sections assume that you have copied the content from the WebStats.zip file onto the host machine and that some web content has been copied to the “www” directory. The “www” directory must be in the same folder in which WebStats is running and must be reachable through the webserver (i.e. your webserver configuration must include the “www” directory to be exposed to the world).

#### 3.1 WebStats.conf

WebStats.conf is the main configuration file of WebStats. The file is well documented, so you cannot really do anything wrong. However, make sure to prepare the following minimal steps to have WebStats ready for deployment:

1. Under “server”, enter the entire path under which a user can reach your web content. This is equivalent to the URL a user would type into the browser.  
Example: server=”<http://www.yourdomain.com/~frank/www/>”
2. Under “server\_name”, enter the URL of your server, without protocol or port.  
Example: server\_name=”[www.yourdomain.com](http://www.yourdomain.com)”
3. Under “port”, enter the port number on which the HTTP server of WebStats shall be reachable.  
Example: port=”4200”

4. Enter the command that you can use to shutdown your HTTP server remotely. This can be left empty if you don't wish to switch off the HTTP server from the web.  
Example: shutdown="/QUIT"  
The server will shutdown when "<http://www.yourdomain.com:4200/QUIT>" is called from a browser
5. Under "db\_url", enter the URL under which JDBC can reach your database  
Example: db\_url="jdbc:mysql://yourdomain.com/WebStats"
6. Under "db\_user" and "db\_pass", enter the user name and password for WebStats that you intend to use with your database. You can keep "db\_pass" empty and provide a command line argument instead.
7. Under "server\_data\_path", enter the directory that you intend to store additional files for the database in. This can be left as it is, if you intend to use the "server\_data" directory from the WebStats.zip file.
8. Under "www", enter the directory that you intend to store your web content in. This can be left as it is, if you intend to use the "www" directory from the WebStats.zip file and have configured your HTTP server to expose this folder to the world.
9. It is recommended not to tamper with the variables "separator" and "escape". Leave those as they are.

### 3.2 webstats.js

This file is located in the "www" directory in the WebStats.zip folder and goes together with your web content and is responsible for collecting client information. To adapt the file, you only need to modify two variables. **Leave everything else unmodified!**

1. Under "server", enter **the same value** that you have entered under "server\_name" in the WebStats.conf file
2. Under "port", enter **the same value** that you have entered under "port" in the WebStats.conf file.

### 3.3 PHP Files

These files can be found in the "php-stats" directory in the WebStats.zip. They provide you with some examples of how to retrieve the recorded statistics from the server. In every file, you need to manipulate three variables:

1. Under "\$db\_host", enter **the same value** that you have entered under

“server\_name” in the WebStats.conf file.

2. Under “\$db\_user”, enter **the same value** that you have entered under “db\_user” in the WebStats.conf file.
3. Under “\$db\_pwd”, enter **the same value** that you have entered under “db\_pass” in the WebStats.conf file. If you choose to keep the “db\_pass” variable without value in WebStats.conf, be sure to enter the same value for “\$db\_pwd” that you provide as the command line argument.

#### 4. Deploying

This section describes how to deploy the application to your webserver. It assumes that you have copied the content from the WebStats.zip file onto the host machine and that some web content has been copied to the “www” directory. The “www” directory must be in the same folder in which WebStats is running and must be reachable through the webserver (i.e. your webserver configuration must include the “www” directory to be exposed to the world).

##### 4.1 Installing MySQL Database

You need to install the WebStats Database into your MySQL server. Do so by opening up a command line and type

```
mysql -p 'yourDB' < WebStats.sql
```

and use for 'yourDB' the database that you intend to use with WebStats (i.e. the same one you references in the “db\_url” variable in the WebStats.conf file). When prompted, type in your DB password.

##### 4.2 Installing WebStats-1.0-bin.jar

Installing WebStats is not necessary. Simply copy the binary file (“WebStats-1.0-bin.jar”) to your host machine in the same directory as the “server\_data” and “www” folders and execute.

##### 4.3 Preparing HTML Files

You need to prepare every HTML file you wish to record statistics for. In the “www” directory from the WebStats.zip archive, you will find two files, i.e. “testpage.html” and “webstats.js”. You need to copy “webstats.js” file into the “www” directory of your host machine and edit **every** HTML file you wish to record statistics for in a similar manner as depicted in the example “testpage .html”. Specifically, you need to do the following:

1. Edit the **header** of the files by adding the following code somewhere in the <head> tag of the HTML file so that it looks like this:

Example:

```
<head>
  <script type="text/javascript"
    src="webstats.js">
  </script>
  <script language="JavaScript"
    src="http://gd.geobytes.com/gd?after=-
1&variables=GeobytesCountry,GeobytesCity">
  </script>
  <script
type="text/javascript">document.write(unescape("%3Cform
name='http://localhost:8080/index.html' action='
id='f0rm' METHOD='POST'%3E%3Cinput type='hidden'
name='h1dd3n1nput' id='h1dd3n' value='j4v4scr1pt'
/%3E%3C/FORM%3E")));
  </script>
</head>
```

2. Edit the **body** tag of your HTML files so that it looks like this:

Example:

```
<body onLoad="run()">
... rest of your body
</body>
```

#### 4.4 Installing PHP Files

To install the sample PHP files and view your recorded statistics most conveniently, simply copy the “php-stats” directory into some place that is exposed to the world wide web (i.e. that is exposed by the web server) on your target machine. You can view your statistics by browsing to the URL of the php files.

Example:

[http://www.youdomain.com/path/to/your/php-stats/sever\\_data.php](http://www.youdomain.com/path/to/your/php-stats/sever_data.php)

## 5. Executing WebStats

Once all of the above steps are completed, your folder in which WebStats will be running should look like this:

```
drwxr-xr-x 6      frank      franksgrp 23      May 5 11:07 www
-rw-r--r-- 1      frank      franksgrp 735029   Apr 30 21:18 WebStats-1.0-bin.jar
-rw-r--r-- 1      frank      franksgrp 1172     Apr 30 21:15 WebStats.conf
-rw-r--r-- 1      frank      franksgrp 445      Feb 22 2007 index.html
drwxr-xr-x 3      frank      franksgrp 3        Apr 30 21:16 server_data
```

You can verify this by using the tool `dir` (in Windows) or `ls -l` in \*UX from command line. Once you have verified that your directory looks like this or similarly (depending on your settings), you can execute WebStats by typing:

```
java -jar WebStats-1.0-bin.jar
```

(assuming you set a password in `WebStats.conf`) or

```
java -jar WebStats-1.0-bin.jar somepassword
```

where `somepassword` is some password for your MySQL database. In Windows, MacOS and some properly configured \*UX operating systems, you can also launch the program by double-clicking on the jar file from some Explorer/Finder/File Browser window of your preference.