

SCRAPER—A PROGRAM TO INSTANTLY CONVERT A STATIC WEBSITE TO
A DYNAMIC WEBSITE

A THESIS

Presented to the Department of Computer Engineering and Computer Science
California State University, Long Beach

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science
Option in Computer Science

Committee Members:

Burkhard Englert, Ph.D. (Chair)
Shui Lam, Ph.D.
Tracy Bradley Maples, Ph.D.

College Designee:

Kenneth James, Ph.D.

By Hemil M. Sheth

B.E., 2006, Vidyavardhini's College of Engineering and Technology

January 2010

UMI Number: 1481772

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

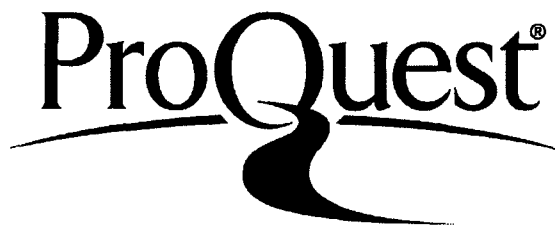
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1481772

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

SCRAPER—A PROGRAM TO INSTANTLY CONVERT A STATIC WEBSITE TO A DYNAMIC WEBSITE

By

Hemil M. Sheth

January 2010

Internet evolved with Web 1.0., as a method to transmit information to the people. As years passed, it became a resource to offer information as well. Soon the era of Web 1.0 ended which gave birth to Web 2.0 with modern rich internet applications. The transition from Web 1.0 to Web 2.0 was the latest development since the use of modern website was not only to store information but also to gather and dynamically interact with people.

The focus of this thesis is to create dynamic website in Joomla Content Management System (CMS) from static HTML pages by using a Scraper program that will parse the entire website and generate a SQL file. The main stakeholders will be the corporate websites, portals, organizations, institutions, companies, et cetera. One can promptly switch from static web pages to dynamic web pages (Web 2.0) which help in managing, storing, revising, publishing and controlling HTML content with its user-friendly interface. One can even access this Joomla! website from any remote computer.

The algorithm used in this creation is slightly complex but straightforward in its implementation.

The Scraper is implemented using PHP functions and regular expressions. The output generated is a SQL file, which contains several insert statements. Thus, the dynamic Joomla! website is generated by importing this SQL file into Joomla!'s database.

The thesis compares the manual way of website conversion and the implementation using Scraper to generate a dynamic Joomla! website. The comparison explains the benefits of using the Scraper. The thesis also focuses on the security issues in the Joomla! website. The conclusion includes the future developments to make this tool more accessible to the users.

ACKNOWLEDGEMENTS

This thesis is the result of two years of work during which I have been accompanied and supported by many people. It is now my great pleasure to take this opportunity to thank them.

First and foremost I would like to thank the ALMIGHTY GOD, the faith I have in him without which I could never have done this.

I am heartily thankful to my supervisor, Dr. Burkhard Englert (*Department of Computer Engineering and Computer Science, California State University Long Beach*) whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of this subject. Also, I would like to thank the Thesis office staff of *California State University Long Beach* for being abundantly helpful, and has assisted me in numerous ways.

I cannot end without thanking my friend Sanyukta Kaza (*Independent Media Production Professional, Chapman University*), family members including my father, mother and sister, on whose constant support and love I have relied throughout my time at California State University, Long Beach.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
1. INTRODUCTION	1
Objective of the Thesis.....	3
Outline of the Thesis	3
Brief Introduction to CMS and Joomla!.....	5
Steps to Install Joomla!	6
Related Work	7
2. SYSTEM TECHNOLOGIES.....	8
Joomla! CMS	8
PHP	10
MySQL.....	10
phpMyAdmin	11
HTML and CSS	11
Structure of Joomla!	12
3. ARCHITECTURE OF JOOMLA!.....	15
Joomla! Framework Overview.....	15
Object	20
Message.....	21

CHAPTER	Page
4. WEB SITE CONVERSION—TRADITIONAL WAY	32
5. JOOMLA! SECURITY	39
Maintaining security using PHP, Apache and MySQL	40
Attacks on Joomla! website	43
6. WEBSITE CONVERSION USING SCRAPER.....	47
Database Architecture	47
Basic Algorithm of a Scraper	50
Creating a Menu Structure	54
7. EXAMPLE OF SCRAPER IMPLEMENTATION	56
Usefulness of the Scraper.....	54
Future developments to the Scraper (Wish List).....	61
APPENDIX: SCRAPER CODE	62
REFERENCES	69

LIST OF TABLES

TABLE		Page
1.	Global Configuration Settings in Joomla!.....	23
2.	Preconfigured .htaccess File Shipped with Joomla!	46
3.	Database Structure of jos_content.....	48
4.	Database Structure of jos_menu.....	49
5.	Database Structure of jos_sections.....	50
6.	Database Structure of jos_categories	51
7.	Comparison Chart between Manual Website Conversion and Implementation using Scraper.....	55

LIST OF FIGURES

FIGURE		Page
1.	Basic web server retrieving data from static HTML file	12
2.	Interaction between web browser and web server running PHP and MySQL via Joomla! CMS	14
3.	Joomla three-tiered architecture.....	15
4.	Traditional MVC pattern.....	16
5.	Advanced Architecture of Joomla! MVC	18
6.	UML's Sequence Diagram showing the logical flow within MVC.....	20
7.	Sample Control Panel Screen.....	22
8.	Default "rhuk_milkyway" template (Joomla template name)	28
9.	Levels in Joomla! Menu Structure	33
10.	Pictorial representation of Scraper Algorithm	51
11.	Screenshot of a Scraper program	57
12.	Screenshot of the SQL file (2541 insert statements) generated by the Scraper	58
13.	Screenshot of a sample static HTML page	59
14.	Screenshot of a sample page in Joomla generated by the scraper.....	60

CHAPTER 1

INTRODUCTION

The internet has become essential in this age; over 1.6 billion of the world's population uses the network for personal, business, entertainment and educational purposes. Websites collectively are the foundation of the internet, just as pages are the foundation of books. In business realm, majority of companies, big and small, have their own domain to share information with the outside world and get feedback as well, therefore the content of a website is vital as it can be browsed by millions around the globe [1].

By definition, a website is made up of individual web pages. A web page is a resource of information written in markup languages such as Hypertext Transfer Protocol (HTML) or EXtensible HyperText Markup Language (XHTML) format, which can be accessed through a web browser. Hypertext Transfer Protocol (HTTP) is a set of rules to transfer information between the web browser and the web server through World Wide Web (WWW).

Now the question arises, how does one maintain these web pages on a daily basis? Creating a website itself takes immense time and effort, and the maintenance of it is an ongoing process. It is relatively manageable to maintain a small website containing only 10 to 20 web pages using Adobe Dreamweaver (Adobe Dreamweaver is software

available to maintain web sites). Such professional web-editors with graphical user interface have made creation and maintenance of a website much easier by minimizing the construction time for a web developer [2]. One can manually go into the web page and edit the HTML code to update it. What if the website expands to more than a few simple pages, then it becomes a complex process to maintain this site even with the support of these advanced web-editors. If a website contains more than 100 *static* web pages, then it becomes a very tedious process to manually update it. In addition, other tasks such as fixing broken links, implementing a menu structure, adding a forum or site search option can make web development an exhausting affair. In order to solve such problems and to make the task easier, one can use the Content management system (CMS) which not only simplifies the process but also makes it accessible to even a non-technical user. This minimizes the problems or inconsistencies introduced into the published content. The implementation of the CMS vanishes the difficulty of website management. In addition to this, features such as site map and search are automatically updated with the latest content. Also, forums, shopping carts and picture galleries are either built into the CMS or easily available through modules or plugins. Thus, with all of the above features it minimizes the time that goes into the development, with reduced number of bugs and security issues. There are several types of CMS's available for use such as Web Content Management Systems (WCMS), Enterprise Content Management Systems (ECMS), Document Management Systems (DMS), Digital Rights Management Systems (DRMS) and Asset Management Systems (AMS) [3].

While choosing a CMS, the factors which need to be considered are, time, type of content—audio, video, speeches, presentations, whitepapers, number of articles and many more. If there is a website which contains 400 to 500 static HTML pages then it becomes impractical to copy and paste the HTML content into CMS. Therefore, automating this entire process is a brilliant solution which can be achieved by studying the sitemap of the website that needs to be converted from static to dynamic. The most essential thing that needs to be studied within the sitemap is the tree structure of the main navigation. After carefully studying the sitemap and the main navigation, a hash table needs to be created to look up the parent and child nodes. Once the parent and their respective child nodes are identified, the process of conversion becomes much simpler.

Objective of the Thesis

The primary objective of this thesis is the development of a Joomla! site using an automated program (Scraper) which is more efficient, powerful and reliable than the manual conversion of the static HTML pages into a Joomla! site.

The other objective of this thesis is to study the architecture, the working of Joomla!, its components, modules, plugins, understanding the power of CMS to build powerful web applications, its merits and demerits.

Outline of the Thesis

Chapter 2 presents a brief introduction to the concepts of CMS, Joomla!, and its importance in the context of free Open source approach to the design, development and distribution of software.

Chapter 3 presents a brief idea of the various system technologies such as Joomla!, PHP, MySQL, HTML, and CSS used to design, develop and deploy the Scraper. Also, a detailed study of the Model-View-Controller (MVC) web application development framework is presented.

Chapter 4 discusses how to convert the static HTML website into Dynamic Joomla! site manually in a traditional way.

Chapter 5 discusses ways to secure a Joomla! site using PHP, Apache and MySQL.

Chapter 6 presents the Scraper algorithm and the implementation functions used to deploy it.

Chapter 7 presents the demonstration and working of the Scraper with sample screen shots.

The main idea of this thesis is to develop, implement and describe an automatic tool that allows a user to migrate a static website into a dynamic Joomla! based website.

The tool will:

- 1). Get all the content from the static HTML website.
- 2). Store it in a format that can be easily used by Joomla! CMS (SQL insert statements).
- 3). Use these SQL queries to create the main body of the Joomla! site.

SQL queries will then create the main menu of the site with the parent nodes, their corresponding child nodes, and the main body for the entire site.

Brief Introduction to CMS and Joomla!

Why CMS?

CMS or Content Management System is a powerful application to manage and control the content of a website from anywhere in the world. It only requires a web browser, internet connectivity and basic Microsoft Word knowledge. In today's world, time is money so the amount of time required to publish the site online is the most important. CMS helps in publishing the new content right away. In addition to this, money is also important criteria while developing a web site. CMS does not require any monthly fees, license fees, maintenance fees [4].

Why Joomla!?

Joomla! is one of the most powerful open source software available on the globe. Joomla! is free software released under the GNU/GPL License [5]. One of the biggest advantages of using Joomla! is that easy-to-use interface. Today, developers are extending the capabilities by creating many modules and extensions. Joomla! not only supports multiple languages and LAMP technology (Linux + Apache + MySQL + PHP) but also supports remote access using a web browser [6, 7].

Joomla! provides free templates and offers low cost development. The templates are a series of HTML, PHP, XML and image files that serve as a basic foundation design within the Joomla! Content Management System. These templates are so helpful for both web programmers as well as the end-users to easily maintain and manage the web content dynamically without any prior knowledge of coding [7].

Despite all these advantages, one question that arises is why has Joomla! not been so popular? This is because people are so accustomed to everyday work of maintaining websites—no matter how obsolete they become—they are often unwilling to switch. Also, switching to this advanced web application takes time and a great deal of effort. Another factor which slows the move to the CMS is the problem to convert the existing hundreds of web pages from their HTML format. This content migration is a challenge but after this initial time devoted to the CMS; it will prove extremely beneficial for the maintenance in the long run [2].

A requirement to use Joomla! as a CMS is the web server which supports dynamic content in the form of PHP and MySQL. Few years ago, availability of such server providers were limited and costly, but times have changed significantly. Nowadays there are numerous web hosting companies with cheap, reliable and secured plans that can run PHP and MySQL. Since, Joomla! has web administered interface, therefore, the server can be located anywhere in the world [2].

Joomla! is used by individuals, corporates, nonprofit and public organizations. An individual with experience in CMS design and website administration can clearly comprehend the reason for Joomla!'s popularity [2].

Steps to Install Joomla!

- 1). Download Joomla! package and upload it on a web server.
- 2). After uploading this package, go to the homepage or the directory where it has been uploaded. For example, www.mysite.com or www.mysite.com/joomla_directory.

3). Prepare the MySQL database. Specify MySQL hostname, database name, username and password.

4). Pre-installation check.

5). The actual installation.

6). Change Joomla!'s administrator password.

7). Change Global configuration settings.

This installation takes about 30 minutes. Once this installation is completed, one can go to the administrative section of the website or browse the website with the sample data. The entire data of this Joomla! website is maintained through a SQL database, which can be administered by phpMyAdmin.

Related Work

Web Content Extractor is also available which serves similar purpose of data extraction from any website [8]. It crawls into the HTML code and extracts the required data. But, compared to Scraper it cannot convert the HTML website into a Joomla! website. The main purpose of the available Web Content Extractor is to extract the content into various formats such as CSV, MySQL, HTML and XML. The Scraper benefits all stakeholders who wish to have an easy, consistent way to do website management.

CHAPTER 2

SYSTEM TECHNOLOGIES

The technologies and the management systems utilized used to generate a Joomla! website are as follows:

- 1). Joomla! CMS.
- 2). PHP.
- 3). MySQL.
- 4). HTML and CSS.

Joomla! CMS

Joomla! CMS is a free, open source CMS written in PHP. It is used to build web applications on WWW as well as on intranets. It uses MySQL database at the back end to store the information. It is released under GNU General Public License. It includes features such as Really Simple Syndication or Rich Site summary (RSS feeds), news flash, blogs, polls, search and multi-language support. It is a robust system and there are several plug-ins available. Joomla! enables to do administrative and maintenance tasks using a simple control panel screen (see Figure 7). It can be managed sitting at home or resting at the beach with a basic Wi-Fi enabled laptop [2]. It has been awarded as the “Best PHP open source CMS” in 2007 [9].

Following are the features of Joomla!

- 1). Website management through a simple and robust web interface.
- 2). Hierarchical user group management.
- 3). Automated menu management including site map.
- 4). Integration with File Transfer Protocol (FTP), email and Light Weight Directory Access Protocol (LDAP).
- 5). Article management by several users at the same time using the lock process automatically so that other users do not overwrite or modify the same content at the same time.
- 6). Moderating remote postings by authors, publishers.
- 7). Built-in “What You See is What You Get” (WYSIWYG) editor.
- 8). Built-in site search of the latest content.
- 9). Built-in Polling, user registration, RSS, blogs, caching for improved performance.
- 10). Optional automatic generation of search engine-friendly (SEF) URLs.
- 11). Multiple language support.
- 12). Accessibility options for the disabled.
- 13). Availability on all major operating systems (Windows, Mac OS and Linux).
- 14). Full open source license with free download of the application and source code.
- 15). Extensions available for shopping cart, picture galleries, inventory management and point of sale [2].

PHP

Hypertext Preprocessor (or PHP) is one of the most popular and powerful open source server-side scripting language generally used for web development to produce dynamic web pages. PHP mainly runs on a web server and is used to output Hyper Text Markup Language (HTML) pages on a web browser [10]. It is platform independent and can be deployed on any web server which has PHP installed. Most stable PHP version is PHP 5.

Advantages of Using PHP

PHP mainly focuses on server-side scripting therefore it is very similar to many Common Gateway Interface (CGI) programs. CGI is a standard for interaction between an external gateway program and the web servers. PHP allows communication between databases such as MySQL and the web browser. Command line scripting and desktop applications can be designed using PHP. It also has the ability to generate EXTensible HyperText Markup Language (XHTML) files, images, pdf files and flash files. It has several built-in functions which help in speeding the entire process. It also supports Object-oriented programming (OOPS) [11].

MySQL

My Structured Query Language (MySQL) is the one of the most popular database used with PHP and also by Joomla! It stores data in a secured manner on a web server and provides access to multiple users at the same time with its robust architecture. It has a variety of features such as stored procedures, triggers, cursors, query caching, indexing and searching, replication support, and many more [12]. It also implements a number of

different storage engines such as MyISAM, InnoDB, BDB, Memory, Merge, Archive, Federated, Cluster/BDB, Other that are particularly designed to handle the application needs.

phpMyAdmin

The phpMyAdmin is a free software entirely written in PHP which handles the administration of one or more MySQL servers over the web [13].

Advantages of Using phpMyAdmin

- 1). Extremely good web interface.
- 2). Supports 57 languages as well as LTR (left to right) and RTL (right to left) languages.
- 3). Supports most SQL features such as view, add, drop, alter, rename databases.
- 4). Maintenance of tables, fields, servers, databases is not a complicated process.
- 5). Manages users and their privileges, stored procedures and triggers.
- 6). Executes batch queries, import, export databases in various formats like CSV, SQL, XML so migration from one SQL to another is simple and easy.
- 7). Supports multiple databases and administers multiple servers.
- 8). Search is made easy in the entire database or a subset of it.

HTML and CSS

HTML is a markup language with well-defined tags understood by a web browser. Cascading Style Sheets (CSS) defines the style and the layout of the text to be presented on the web. HTML code is embedded into the PHP script in order to generate a dynamic web page which is the foundation of Joomla! CMS. It defines various data types such as

IDs, names, numbers, languages, colors, character, date, time and so forth. The combination of HTML, CSS and scripts make *Dynamic HTML* which is also known as “Document Object Model” [14].

Structure of Joomla!

Joomla!’s structure can be divided into two sub-structures: first contains the actual PHP files and the second contains the database in the form of MySQL. Therefore, the website is much lighter in weight meaning, the size of the entire website is very small.

Figure 1 shows an interaction of a web browser with a simple web server. When a user requests a HTML page from a web server, the server simply retrieves the HTML code from the static HTML file as it is and then returns it to the web browser. The HTML code returned to the web browser is called static web page since the content contained in the file stored on the server is exactly the same without any revision [2].

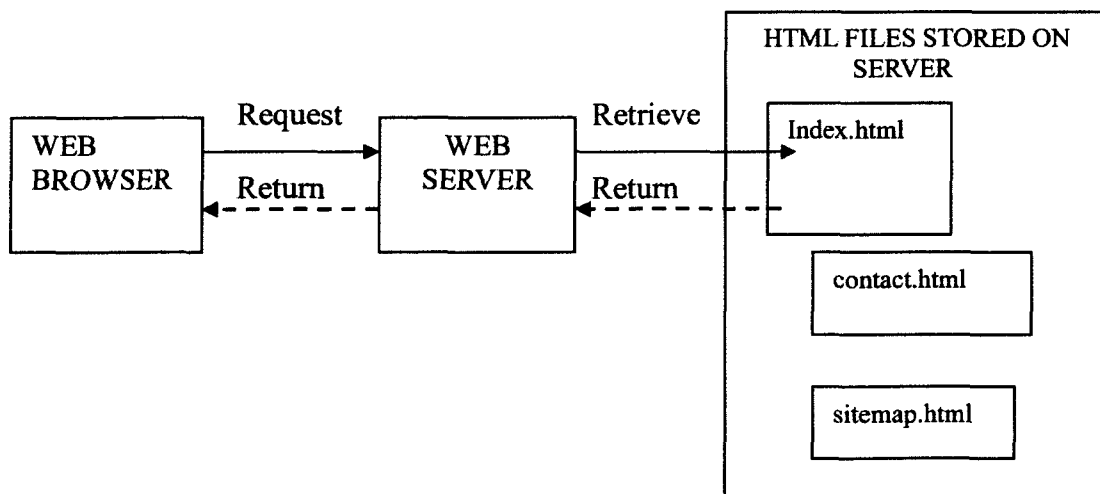


FIGURE 1. Basic web server retrieving data from static HTML file [2].

In Joomla! CMS, the above process is executed in a similar fashion. Figure 2 shows an interaction of a web browser with a web server running PHP and MySQL. When a web browser makes a request to a PHP page from a web server, it actually activates the server's PHP engine through Joomla! It analyzes the request and fetches the content from MySQL's database server (see Figure 2). Once the content is retrieved, Joomla! formats the content and then returns to the browser as a static HTML page. The skin of the Joomla! website can be changed by just changing its template from the administrative back-end.

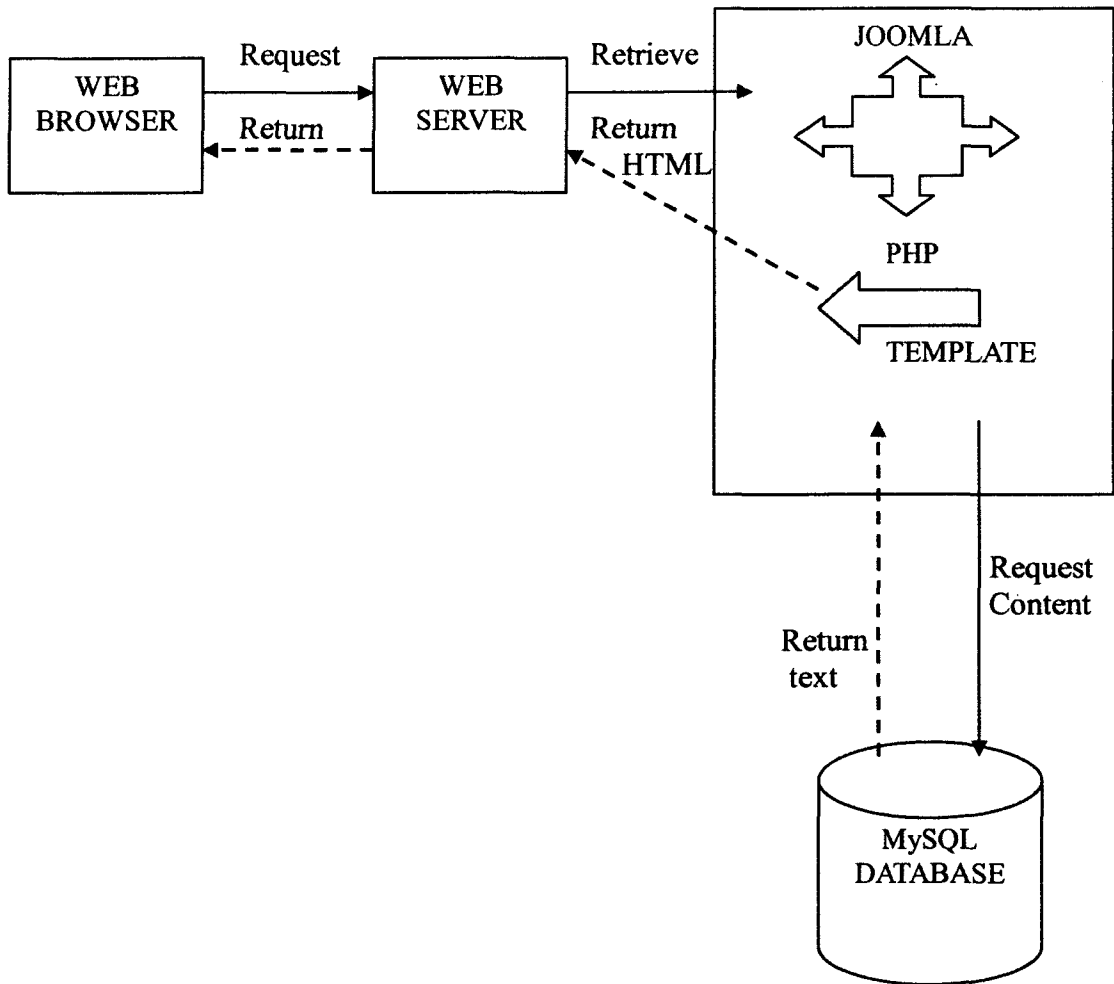


FIGURE 2. Interaction between web browser and web server running PHP and MySQL via Joomla! CMS [2].

CHAPTER 3

ARCHITECTURE OF JOOMLA!

Joomla!'s Framework is the foundation of the Joomla! CMS.

Joomla! Framework Overview

Joomla! has three-tiered architecture as described below [15] (see Figure 3):

- 1). The bottom Framework layer—containing libraries and plugins known as mambots.
- 2). The middle Application layer—containing three applications: JInstallation, JAdministrator and JSite.
- 3). The Top Extension layer—contains all components, modules and the template logic is executed and rendered.



FIGURE 3. Joomla! three-tiered architecture. (Reproduced with permission of Open Source Matters Inc. in print and electronic versions with copies sold on demand by the publisher ProQuest/UMI.)

Model-View-Controller Web Application Framework

Joomla! components are built-in Model-View-Controller (MVC) pattern. MVC is a design pattern to separate business logic and data presentation. The basic idea to separate them is to group business logic into a separate group so that the data, which is bounded by the interface and the user interaction, can be revised and modified without recoding the business logic. MVC was initially developed to represent the traditional input, processing and output roles into a logical GUI architecture as show in the diagram below:

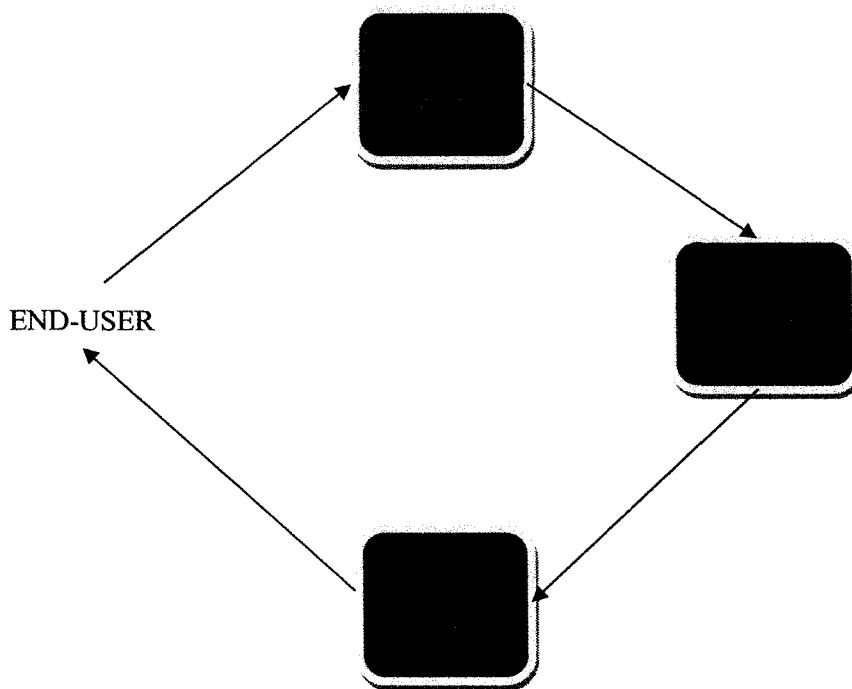


FIGURE 4. Traditional MVC pattern [16]. (Reproduced with permission of Open Source Matters Inc. in print and electronic versions with copies sold on demand by the publisher ProQuest/UML.)

MVC pattern is systematically organized into three distinct roles [16]:

1). Model: This encloses the application's data, application flow and business logic. It provides ways to control and manage this data in a very significant way. The model contains functions to update and make changes to the database. If application is to be moved from one system to another, then only the Model component needs to be altered without touching the view or the Controller. This helps to expand and revise the properties of one section without changing other sections. This is the main advantage of using MVC. This saves a lot of time and money because one can reuse the same pattern for different applications.

2). View: The View extracts the data from the Model and formats it in a manner, which is suitable for interaction. For example, an HTML page would be a View for a web application. In this component, the data is neither modified nor updated; it only displays data that is retrieved from the Model. The data is then passed to the Controller and is available for the end user to view.

3). Controller: This controls the user actions by determining what request the user is making and responds by activating the Model to manipulate the data and pass that data into the View. For example, in a web application, the user makes a web page request. This request is passed to the Controller that takes action and passes the data into the Model, where the data is modified or updated and then passed on to the View, where the template is added so that the user can view the data in a presentable format. Figure 5 shows the advanced architecture of Joomla!'s Model-View-Controller.

JOOMLA MODEL VIEW CONTROLLER (MVC)

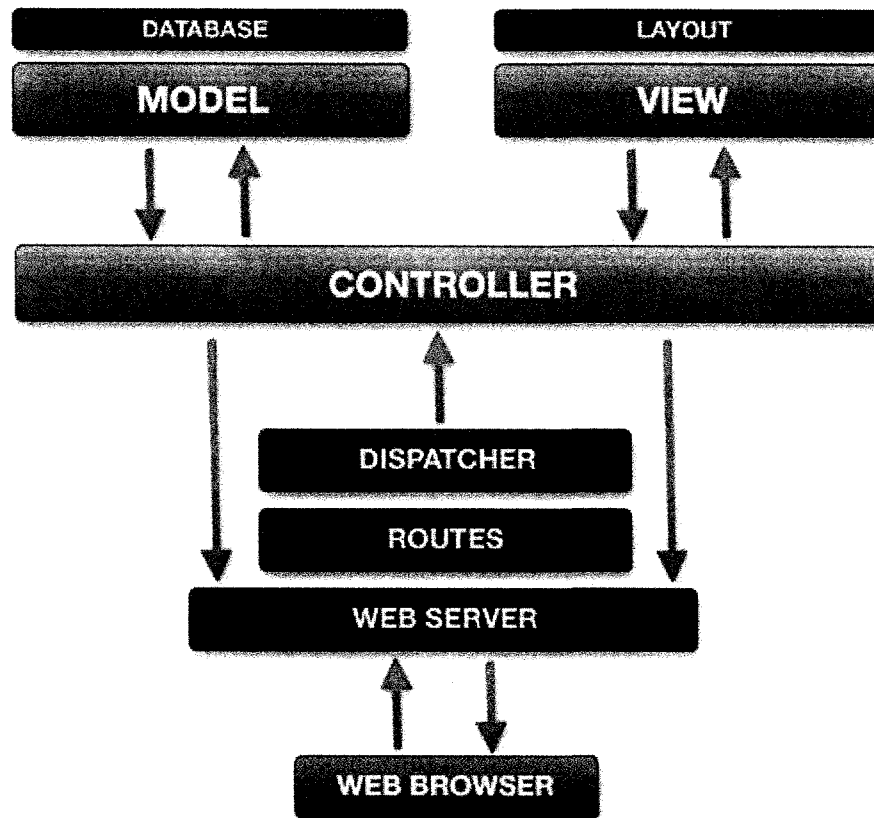


FIGURE 5. Advanced Architecture of Joomla! MVC [7]. (Reproduced with permission of Open Source Matters Inc. in print and electronic versions with copies sold on demand by the publisher ProQuest/UMI.)

In order to implement the MVC pattern, the following modules are created:

- 1). Component.
- 2). Controller.
- 3). View.
- 4). Template (XML file).
- 5). Model.

Each element of MVC component is simple and it provides a great flexibility and power using Object oriented programming (OOP). In this way, the MVC component helps in retrieving data from the table in the database.

How MVC Pattern works in Joomla!

We will now discuss how this pattern works, that is, the logical flow of communication within the Model-View-Controller. In order to understand the flow, one has to know what the entry point is. UML's Sequence Diagram (See Figure 6) is the best way to represent the flow within the system [17]:

Sequence Diagram represents the logical flow within a system in a visual way so that one can analyze and design the system. It shows the actions taken within a system in a systematic way. It is also the best way to represent the dynamic behavior of the system. Sequence diagram is made up of objects and messages. It is shown in 2-D with X-axis representing the lifeline of an object and Y-axis representing the sequence of creation of the objects [17].

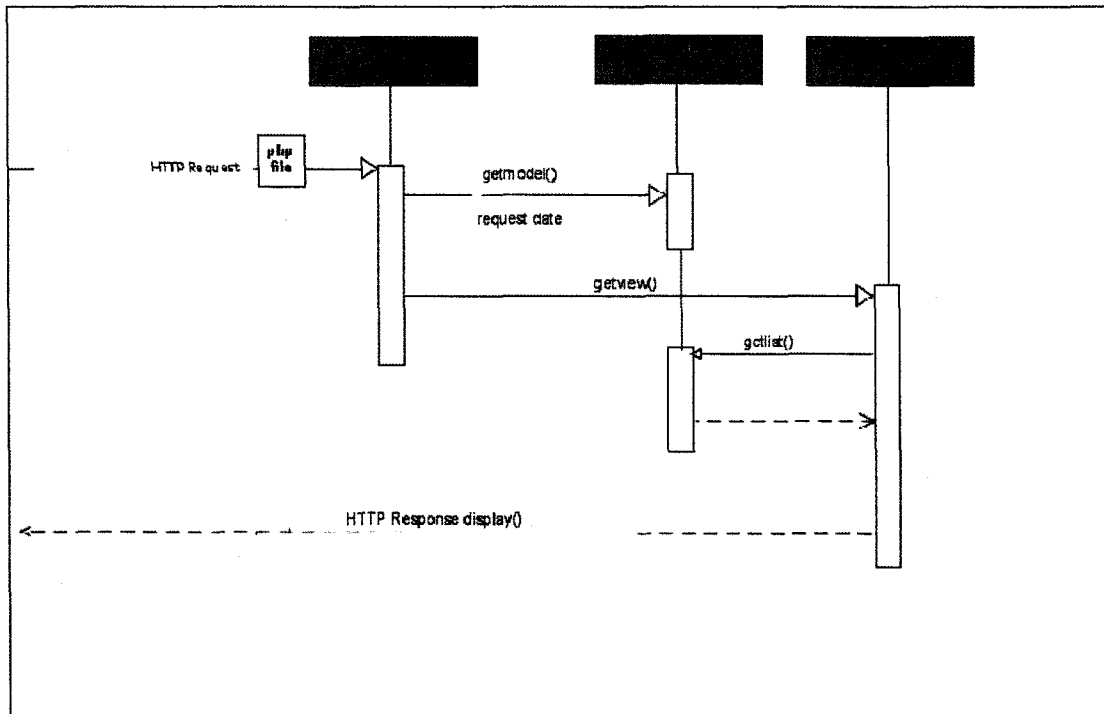


FIGURE 6. UML's Sequence Diagram showing the logical flow within MVC [17].
 (Reproduced with permission of phpeveryday.com in print and electronic versions with copies sold on demand by the publisher ProQuest/UMI.)

The elements of the sequence diagram in the above scenario (see Figure 6) can be defined as follows:

Object

Object is the primary element in a sequence diagram. It is an instance of a class and is represented by a rectangle. Following are the objects in the above scenario (see Figure 6):

- 1). Controller.
- 2). Model.
- 3). View.

Message

The interface between different objects is represented as a message. It is represented by a directed arrow in the direction in which the message is sent.

In the above scenario, there are different messages such as

- 1). *Request date*: Controller sends a request to the Model.
- 2). *Get view*: Controller sends a request to the View.
- 3). *Get list*: View sends a message to the model and in turn Model sends back a reply message to the View which is represented by a dotted directed arrow.
- 4). *Response*: View sends a reply message back to the user and displays the requested data.

In this scenario, the end user makes a HTTP request through a PHP file. This tells the Controller to invoke the `getmodel()` function to request the date from the Model through `all.php` file.

After sometime the controller sends a signal “`getview()`” to View through `view.html.php` file. As soon as the View is invoked, it results in sending a signal to the Model to get the list of data via `getlist()` function. In return, the Model replies the View by sending a response, which in turn returns the result to the end user and displays the data on the screen. This is how the Model-View-Controller works within the system.

Joomla!’s Components

Joomla!’s architecture is similar to any other website. Joomla!’s administration is managed through the back-end control panel as shown in Figure 7. Once the administrator logs into the Joomla! website from the back-end, the first screen viewed is

the Control Panel. From the Control Panel one can create and manage articles, sections, and categories. Super-Admin can also manage the Media, Menu, Language, User, Module, Plugin, Template and the website's Global Configuration in the Control Panel. Joomla!'s Super-Administrator has privileges to change the following settings in the Global Configuration.

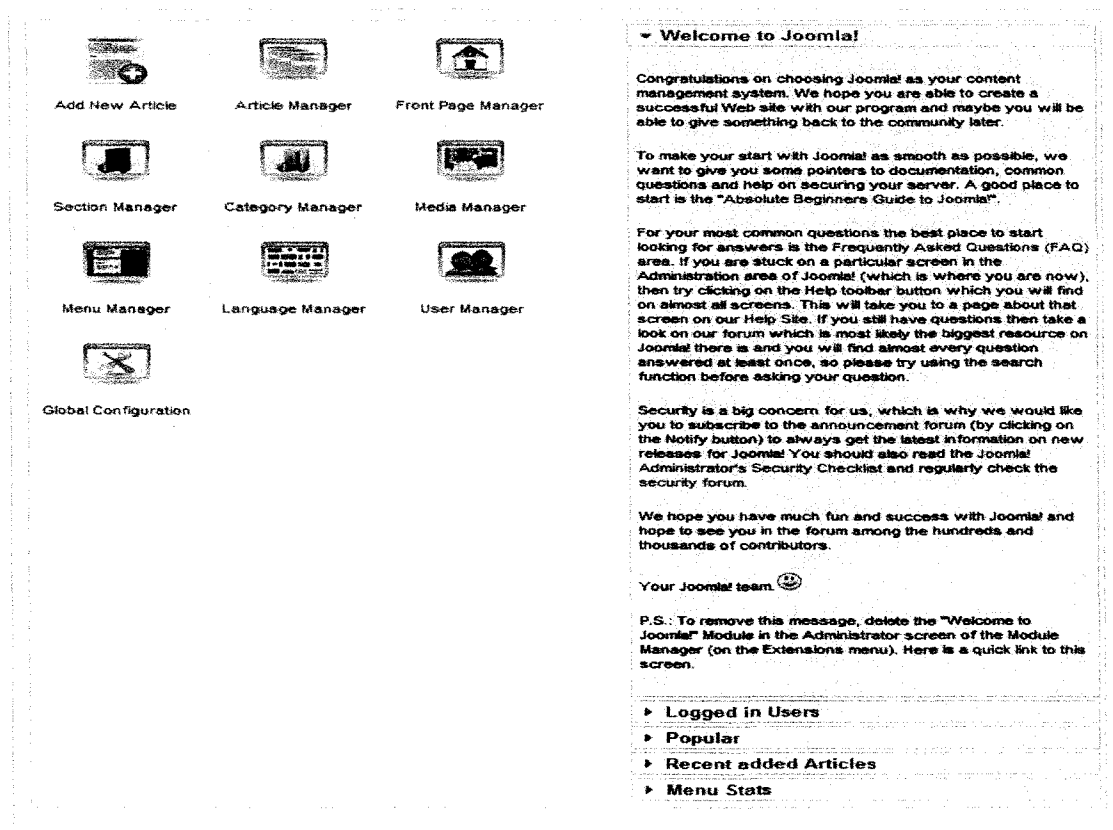


FIGURE 7. Sample Control Panel Screen [18]. (Reproduced with permission of Open Source Matters Inc. in print and electronic versions with copies sold on demand by the publisher ProQuest/UMI.)

TABLE 1. Global Configuration Settings in Joomla! [18]

<u>Settings Name</u>	<u>Description</u>
<i>Site Settings</i>	
1). Site Offline	Select whether access to the Site Front-End is available.
2). Offline Message	A message that displays if the site is offline.
3). Site Name	Enter the name of the site.
4). Default WYSIWYG Editor	Select the default WYSIWYG editor.
5). List Length	Sets the default length of the list in the Control Panel for all users.
6). Feed length	Select the number of content items to show in the feed.
7). Feed Email	The RSS feeds include Author's email address. Select Author's Email or site's email address to include the "Mail From" for each article.
<i>Metadata Settings</i>	
1). Global Site Meta Description	Enter a description of the overall Website that is to be used by search engines.
2). Global Site Meta Keywords	Enter the keywords that best describe the Website.
3). Show Title Meta Tag	Show the Title Meta Tag when viewing Articles.
4). Show Author Meta Tag	Show the Author Meta Tag when viewing Articles.
<i>SEO Settings</i>	
1). Search Engine Friendly URLs	Select whether or not the URLs are optimized for search engines.
2). Use Apache mod_rewrite	Select to use Apache Rewrite Module to catch URLs that meet specific conditions and rewrite them as directed.
3). Add suffix to URLs	If yes, the system will add a suffix to the URL based on the document type. It will add .html at the end of the URL.
<i>System Settings</i>	
1). Secret Word	This is an auto-generated, unique alphanumeric code for every Joomla! installation. It is used for security functions.
2). Path to Log folder	To create logs in Joomla! specify a path to a folder

TABLE 1. Continued

<u>Settings Name</u>	<u>Description</u>
3). Enable Web Services	Enable the ability of installation to make Remote Procedure Calls (RPC) using HTTP as the transport medium and XML as the encoding language. This function is required to ensure that many third-party extensions work. Default is No.
4). Help Server	Select the name of the Help Server from which the system will collect the Help Screen displays.
<i>User Settings</i>	
1). Allow User Registration	If set to Yes, new users allowed to self-register.
2). New User Registration Type	The default access level that will be applied to new users registering via the front-end.
3). New User Account Activation	If set to Yes, the User will be e-mailed a link to activate their account before they can log in.
4). Front-end User Parameters	If set to Show, users will be able to select their language, editor and Help Site preferences on their Details screen when logged into the front-end.
<i>Media Settings</i>	
1). Legal Extensions (File Types)	Extensions for the files allowed to be uploaded.
2). Maximum Size (in bytes)	Maximum size for an upload (in bytes).
3). Path to Media Folder	Path to the folder to use with the Media Manager.
4). Path to Image Folder	Path to the folder to use with the Image Manager.
5). Restrict Uploads	Restrict uploads for lower than manager users to just images.
6). Minimum User Level for Media Manager	Select the lowest user level that can use the Media Manager in the front-end.
7). Check MIME Types	Use Multipurpose Internet Mail Extensions (MIME) Magic or Fileinfo to attempt to verify files.
8). Legal Image Extensions (File Types)	Allowed Image Extensions to upload.
9). Ignored Extensions	Ignored file extensions for MIME type checking and restricted uploads.
10). Legal MIME Types	A comma separated list of legal MIME types for upload.
11). Illegal MIME Types	A comma separated list of illegal MIME types for upload.

TABLE 1. Continued

<u>Settings Name</u>	<u>Description</u>
12). Enable Flash Uploader	Select whether the Flash Uploader should be used or not for uploading media in the Media Manager.
<i>Debug Settings</i>	
1). Debug System	If enabled, diagnostic information, language translation and SQL errors will be displayed.
2). Debug Language	Select whether the debugging indicators (‘...’) or (?...?) for the Joomla! Language files will be displayed.
<i>Cache Settings</i>	
1). Cache	Select whether the cache is enabled or not.
2). Cache Time	The maximum length of time in minutes for a cache file to be stored before it is refreshed.
3). Cache Handler	The default caching mechanism is file-based.
<i>Session Settings</i>	
1). Session Lifetime	Auto log out a User after they have been inactive for the entered number of minutes.
2). Session Handler	The mechanism by which Joomla! identifies a User once they are connected to the website using non-persistent cookies.
<i>Server Settings</i>	
1). Path to Temp-folder	Please select a writable Temp folder.
2). GZIP Page Compression	Compress buffered output if supported.
3). Error Reporting	Select the appropriate level of reporting from the drop down list.
4). Force SSL	Force site access to always occur under SSL (https) for selected areas.
<i>Locale Settings</i>	
1). Time Zone	Current date and time configured to display.
<i>FTP Settings</i>	
1). Enable FTP	Enable the built-in FTP functionality of Joomla! to be used instead of the normal upload functionality of Joomla!
2). FTP Host	Enter the name of the host of the FTP server.

TABLE 1. Continued

<u>Settings Name</u>	<u>Description</u>
3). FTP Port	Enter the port that FTP should be accessed by. Default value is Port 21.
4). FTP Username	The username used to access the FTP server.
5). FTP Password	Enter the FTP password.
6). FTP Root	The path to the root directory of the FTP server.
<i>Database Settings</i>	
1). Database Type	The type of database in use.
2). Hostname	The hostname for the database.
3). Username	The username for access to the database.
4). Database	The name of the database.
5). Database Prefix	The prefix used for the database. By Default, it is "jos_"
<i>Mail Settings</i>	
1). Mailer	Select which mailer to use for the delivery of the site e-mails.
2). Mail from	The e-mail address that will be used to send site e-mails from.
3). From Name	By default, this field is populated with the site name entered during the installation.
4). Sendmail Path	Enter the path to the sendmail program directory
5). SMTP Authentication	Select yes if the SMTP Host requires SMTP authentication.
6). SMTP Username	Enter the username for access to the SMTP host.
7). SMTP Password	Enter the password for access to the SMTP host.
8). SMTP Host	Enter the name of the SMTP host.

Joomla!’s website has content, navigation, functionality, color/design scheme, but the organization is different in each section.

1). *Content*: MySQL database handles all the content in Joomla! CMS. The content is called as Articles in Joomla! It is managed through Article Manager. Further,

the Articles can be organized into Sections and Categories. In addition, Sections and Categories are independent from the navigation of the Joomla! website. The main advantage of using Joomla! CMS is that any number of people can add the content without dealing with the format, font, and color changes [15].

a). *Sections*: They are at top-level in the classification. Using Section Manager in the back-end, one can edit the existing section of the website or add a new one. Sections are used to organize Articles. Joomla! has a built-in section named “Uncategorized.” All the articles are assigned either to an already created section or to an Uncategorized section. Super-Admin can decide the access level to this section, that is, who can access this item. There are 3 access levels available in Joomla!

i). **Public**: The Articles are visible to all;

ii). **Registered**: The Articles are only available to Registered users of the Joomla! website.

iii). **Special**: The Articles are only available to Authors or higher status such as Editor, Publisher, Manager, Admin, or Super-Admin.

b). *Categories*: Categories are the second level below the Sections. Several categories can be under one Section. Joomla! has a built-in category called “Uncategorized” Category under “Uncategorized” Section. An already created Section is assigned to the new Category.

One Article can only be in one Category and Section. If the Joomla! website contains a large number of articles then it is easy to group the articles into the Sections and Categories using Article Manager so one can find them without difficulty.

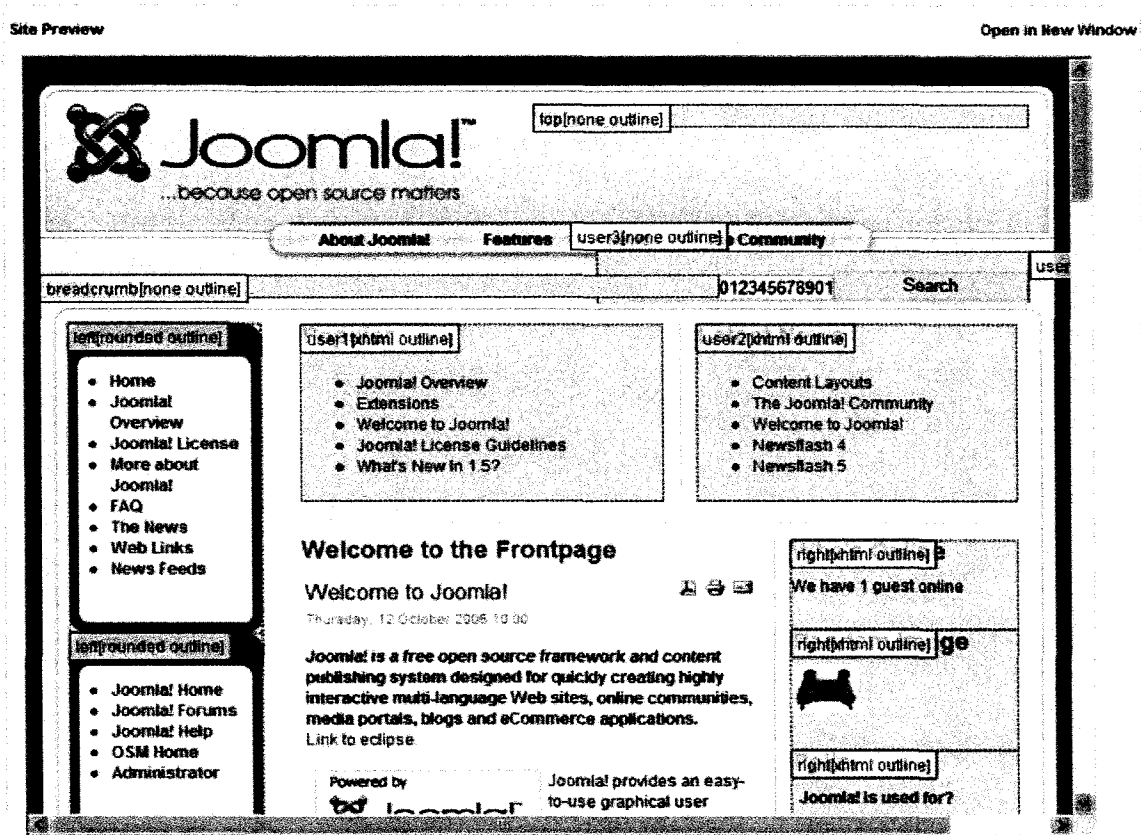


FIGURE 8. Default “rhuk_milkyway” template (Joomla! template name). (Reproduced with permission of Open Source Matters Inc. in print and electronic versions with copies sold on demand by the publisher ProQuest/UMI.)

2). *Templates*: This is the most significant and the toughest part that decides the entire website’s look and feel. This is managed through the Template Manager. The Template Manager assigns a default template to the Joomla!’s website. One can also edit and preview the templates here. Templates contain the style and the layout information that tells Joomla! exactly how each page of the website should appear. It involves coding in PHP that has several pieces of code to fetch the HTML content pages. All the scripts and CSS are loaded through the Template. CSS handles the entire website’s font, color,

background, position, et cetera. One can use the same template for all the pages of the Joomla! website or can assign different templates for each one. One can install or uninstall through Template Manager. Also, Joomla! provides different templates for front and back-end. This is how the front-end “default rhuk_milkyway” template (see Figure 8) looks like which is shipped with Joomla! One can edit the CSS and HTML from the back-end to change the template.

3). *Navigation*: In Joomla!, Navigation is handled through Modules. It contains the most important menu item called Main Menu. Each Menu in the Main Menu has a content item associated to it. A Module is a block of content items. A module can be applied to different pages around the website. Every Menu is accompanied by a Menu Module. Some Modules are linked to Components. Each Joomla! installation is accompanied with 20 built-in Modules in the front-end and 16 built-in Modules in the back-end of the website. The Menu Module is used in every Joomla! website [15, 18].

4). *Functionality*: Functionality is the capability to serve a purpose well. Modules and Components in Joomla! add the functionality to the website. Examples of Components in Joomla! are WYSIWYG editors, polls, newsfeed, search, et cetera. Modules are Breadcrumbs, RSS feeds, footer, main menu, et cetera. One can also install or uninstall various Modules and configure them according to the website Requirements [18].

5). *Front Page Manager*: This controls the pages displayed on the front page of the Joomla! website along with its order [18].

6). *Media Manager*: This manages the media files such as images, documents of the Joomla! website. Here, one can upload or delete files. The default image directory in Joomla! is {site root}/images/. One can change this from Global Configuration but it is not recommended to change it. Also, it offers two views: the thumbnail view which shows the image preview and the details view which shows the file name, dimensions and file size. One can also create sub-directories inside the images folder [18].

7). *Menu Manager*: This manages how the Joomla! menus will look and perform. The Main Menu is always the default menu item because it contains the home page, therefore, the default menu should not be deleted in any case. A menu item can be a blog, table or list. There can be several menus in the website, each with unique name, title and description. Only alphanumeric characters without spaces are allowed as a unique menu name. Each menu can have several menu items within it. When one clicks on the menu item, it directly takes the user to the Menu Item Manager for that selected menu. One can add or edit the menu items in the Menu Item Manager. Here, the access level (Public, Registered or Special) of the menu is decided [18].

8). *Language Manager*: This manager lets one to set the default language for both front and back-end of the Joomla! website. One can install any language from the set of available languages for installation. So the Joomla! CMS generates that particular language selected by that particular user. This does not affect the content items of the Joomla! website. One can set different front-end and back-end languages. Individual Articles can also be configured to use a different language through the Advanced Parameters [18].

9). *User Manager*: This menu shows a list of all the users of the Joomla! website with the ability to sort them via names, usernames, groups, e-mails, last visits. One can also edit or create a user through the user manager. Also, one can see the last login date and time of the user. The following users group is available:

a). *Guest User*: They are Anonymous users of the website with no special rights.

b). *Registered User*: They are normal users who register to the website to view the registered sections of the website. They cannot edit or submit articles.

c). *Author*: They can only submit new articles but cannot edit existing articles after getting approved from the Publisher or a higher person.

d). *Editor*: They can submit, edit articles from the front-end only after getting approval from the Publisher or a higher person.

e). *Publisher*: They can edit, submit or publish articles from front-end only.

f). *Super-Admin, Admin, and Manager*: They have all of the above plus can log into the back-end with increasing rights [18].

10). *Global Configuration*: The Global Configuration panel controls Joomla!'s operational settings. The configuration.php file is updated if any changes is made in Global Configuration. This is useful because one can change these settings as explained in detailed in Table 1.

The main advantage of using Joomla! is its flexibility to accept any design. It gives too many options to approach one thing in different ways.

CHAPTER 4

WEBSITE CONVERSION—TRADITIONAL WAY

In this section, we will see how an existing static HTML website is converted into a Joomla! website.

Steps of conversion:

1). An article is added manually for each page of static content of the existing website in the article manager. So one will have ‘n’ articles for ‘n’ static HTML pages.

2). If there are a few static pages in the existing website, then the articles are set to the built-in Category and Section named “Uncategorized.” If one has more pages than one has to create sections and categories respectively according to the article content. This helps to edit the articles more easily instead of going through the entire Article Manager.

3). Usually the HTML content is copied and pasted to create articles in the Article Manager. Depending upon the custom HTML page used, one can copy paste directly from the browser into the Joomla!, else, one has to copy the HTML into plain text editor and then paste it into Joomla!

4). Navigation in Joomla! is done through the Menu Manager. Now one will have all the articles ready to be attached to the menu. If the existing website has a Site Map, one can easily create the main menu items using menu manager.

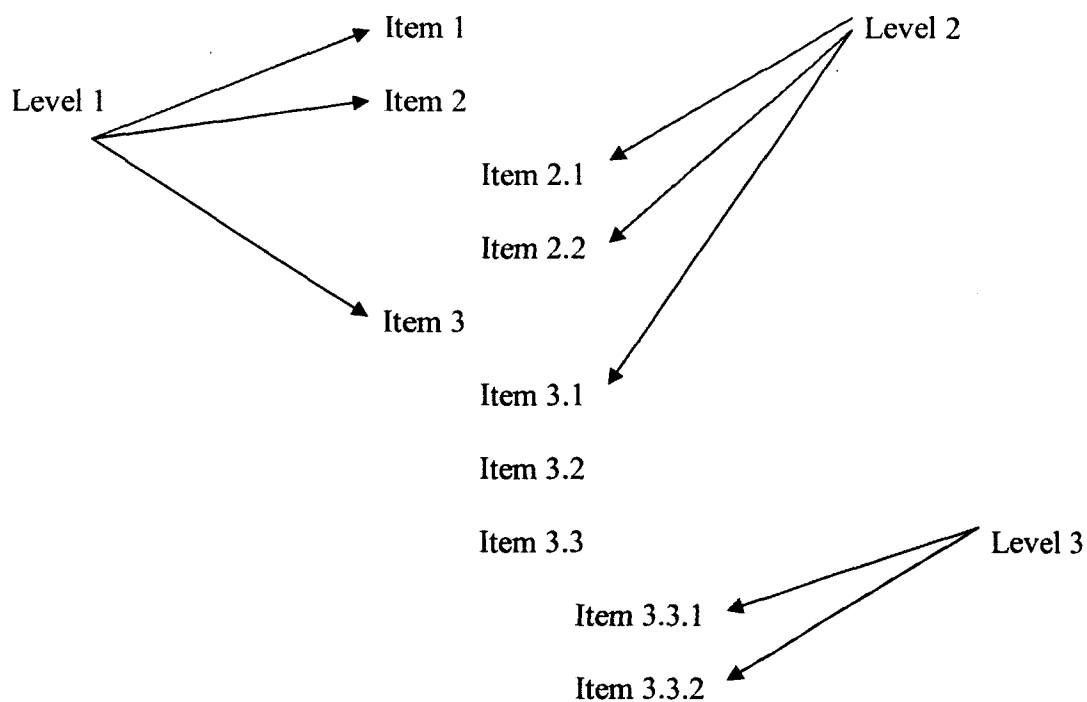


FIGURE 9. Levels in Joomla! Menu Structure.

The menu structure can have as many levels as required as shown in Figure 9. All the menu items are attached to their respective articles. One can also create a customised menu item, or a customised module which helps the user to navigate the website easily [18].

Joomla! offers flexibility to display the content on the web with many layout formats such as article layout, category blog and category list layout, section blog and section layout, front-page blog and blog archive layout.

Article layout: This layout displays a single static content on the page.

Category/Section Blog layout: A blog layout in Joomla! has 3 areas: Leading, Intro and Links. Leading area always uses one column, using full display width. Intro Area may display one, two or more columns depending on the settings. Advanced parameters control the number of articles in each area along with the order of display. Article with “Read More...” break will only display the Intro text along with “Read More...” link. The link area shows the number of links. This layout displays a listing of all the content items of a particular category or section. It shows the title and the introduction text of each content article in that particular category or section. Customization of this layout is done through content parameters.

Category List Layout: This layout is usually used to view a list of all news feed categories. It has the basic parameters such as news feed description which can be viewed or hidden; news feed description text; image and image alignment.

Section Layout: This layout is used to show articles by category in one section. It shows a list of categories in the section. Each category then links to each article in that category similar to category blog layout. It has some basic and advanced parameters. Basic parameters include the sections which need to be selected with descriptions and images; category list section which can be viewed or hidden; a choice to show or hide empty categories that contain no articles; show or hide number of articles in each category with category description. Advanced parameters include:

1). Category order with possible options such as article ordering only by primary order, without regard to category; alphabetical ordering of categories; reverse alphabetical ordering of categories; normal category order as in category manager.

2). Primary order with possible options such as default, oldest first, most recent first, alphabetical title, reverse alphabetical title, alphabetical author name, reverse alphabetical author name, most hits, least hits, order of articles as in article manager.

3). Show or hide a RSS feed link.

Front Page Blog Layout: This layout is similar to Category/Section Blog layout with 3 areas: Leading, Intro and Links.

Archived Blog Layout: This layout shows a list of articles that have been archived and can be date searchable. The articles contained in uncategorized section will not be shown in the archived blog layout. It has basic parameters of sorting articles.

Internal Contacts link: This menu type contains two layouts for displaying contact information.

Contact Category Layout: This layout is used to show all of the published contacts in a given category.

Standard Contact Layout: This layout is used to show contact details of a person. The parameters include address, email, telephone, mobile, fax, miscellaneous icons, contact's position with telephone, mobile and fax number, and so on.

Internal Polls link: This menu type expands to display the results of a poll layout.

Internal Search link: This menu type expands to display the results of the search layout. It has basic parameters which include whether or not to allow a user to limit the

search to any combination of articles, web links, contacts, categories, sections and newsfeeds and show or hide the created date and time of the article. It has the following options under Parameters–Component:

- a) Whether or not to gather the search statistics.
- b) Whether to show or hide the created date and time of the article.

Internal User link: This menu type has six layout options for display.

Default Login Layout: This layout allows the user to logon to the website with the basic login form with creating a new account and Forgot Username/Password links. The basic parameters include: login page title; the URL of the page that the user will be directed to after login; optional login message; optional login description; optional logout page title; logout redirection url; optional logout message and description; and so on.

Default Registration Layout: This layout includes a form with text fields for Name, Username, E-mail Address and Password.

Default Remind Layout: This layout allows the user to receive the username in their email in case they forget their username.

Default Reset Layout: This layout allows the user to reset their password. Here a token is sent to the email address provided which helps in resetting their password.

Default User Layout: This layout shows a welcome message when a registered user logon to the Joomla! website.

User Form Layout: This layout allows the user to change their account details, password, front-end language, text-editor, help site and time zone.

Internal Web links: This menu type has three layout options for display:

1). *Category List Layout*: This layout shows all the web links in a category with optional RSS feed link.

2). *Web Link Category List Layout*: This layout shows all the web link categories. The basic parameters include the image for that page with an alignment and a link for RSS feed.

3). *Web Link Submission Layout*: This layout shows a form to allow the Authors, Publishers, or Editors group to submit a web link through the front-end of the website. The parameters include Name, category, URL and description.

Internal Wrapper link: This menu type is used to display an external website into the Joomla! website using HTML iframe (An inline frame that places another HTML document in a frame inside a normal HTML document). Navigation is possible in the wrapped website which is contained within the Joomla! website. The basic parameters include the URL of the website, optional scroll bars—horizontal or vertical, width and height of the wrapped website.

External link: This menu type is used to create a menu item that links to an external website. It has one basic parameter to display an image to the left or right of the menu item.

Separator: This menu type creates a separator or placeholder within a menu which is normally used to break up a long menu. It has one basic parameter to display an image to the left or right of the menu item.

Alias: This menu item type creates an alias to an existing menu item so that one can have identical menus on two or more different menus without duplicating the

settings. If one changes a parameter of the original menu item than the alias linked to it will automatically acquire the same change.

Custom components installed: Joomla! Super-Administrator can install any components, modules or plugins according to the requirements of the website. There are numerous number of free and paid extensions available to install which makes a Joomla! website more functional [18].

CHAPTER 5

JOOMLA! SECURITY

Security always comes first when it comes to the internet and it was always a concern in the past days and remains the same to date. As new techniques and technologies emerge; security is also a new challenge. There is no one right method to make a website secure and all the security approaches are subject to enhancement, amendment and obsolete at any point of time. Web security is not always guaranteed. An individual with experience is necessary to maintain a secured Joomla! website. While securing any website, the following guidelines should be practiced [18].

Backup early and often: A back up process of the files and the database of the website should be done regularly whenever any change is made in the content or the structure of the website. This helps one to recover from almost any possible failures or attacks [18].

Update early and often: One should always update to the latest stable version of the CMS or any installed third-party components, modules or plugins. This step ensures that the website is protected from the latest vulnerabilities such as content spoofing, SQL injection, information leakage, HTTP Response splitting, cross-site scripting and from all the cyber attacks [18].

Using a secure host: This is the most essential guideline as the use of a high-quality host makes a lot of difference in ensuring proper web security [18].

Maintaining security using PHP, Apache and MySQL

1). Apache's .htaccess (distributed configuration) file should be used to block attack attempts to a website. Using this file, one can protect sensitive directories and files by restricting it with passwords or certain IP addresses. Also, it is recommended to use PHP5 instead of PHP4 as it is used to override the default server settings (see Table 2) [19, 20].

Understanding the .htaccess File

a). *Options + FollowSymLinks:* It plays an important role in web server security. It tells the web server to follow Symbolic links [9]. In order to best explain the concept of SymLinks, let us take an example; suppose a user browses a website, by default the website is set up to show pictures and content. But they may be physically located on a different server. One will find that there are some relative links to the images or the content items when the page source is viewed.

For example, ``. When one browses this image link into the web browser one will see the image but when one tries to physically locate the link on the web server by going into the folder `system/images/stories/`, one may not find it. It is at this point where one has configured the symbolic link. This image actually resides in some other folder; let us say for example in `/files/pics/October_pictures/example.png`; thus, the Symlink tells the web server that whenever the user requests for `/system/images/stories/example.png` then show

/files/pics/October_pictures/example.png. So, in case of Joomla!, FollowSymlinks deals whether it should or should not follow the symbolic links. If the settings of FollowSymlinks is disabled, then browsing to the relative link would return either 404 (not found error) or 403 (access forbidden) error. By default, this setting is enabled in Joomla!

b). *Rewrite engine ON*: By default, this is OFF in Joomla! This setting allows Joomla! to use the `mod_rewrite` settings of the Apache's web server. This creates Search Engine Friendly(SEF) URL's. For example, in Joomla! the URL *www.hemilsheth.com/index.php?option=com_content &view=articleid=1* is tough to understand. So, if `mod_rewrite` is enabled then this URL is converted to *www.hemilsheth.com/webalias* which is very simple to understand for the search engines and human beings.

c). *RewriteBase '/'*: This means that the base directory of the website is rewritten to '/' in order to properly navigate around the website [19].

2). Using Apache's "`mod_security`" and "`mod_rewrite`" filters, one can protect the websites from PHP attacks.

3). Access to MySQL accounts should be cautiously configured and limited with strong usernames and passwords.

4). `Php.ini` files should be configured correctly. It should be copied in the Joomla! root directory and also in the administrator directory.

5). As PHP has a lot of potential to mess the server and even hack the user accounts and get the root directory so the dangerous PHP functions should be disabled

using “disable_functions”. Typical setup include: disable_functions = show_source, system, shell_exec, passthru, exec, phpinfo, popen, proc_open.

6). PHP’s open_basedir should be enabled and configured appropriately since it is meant to restrict scripts to access files outside a set of configured base directories. Before the actual call is performed, the check for this are placed within the PHP functions.

7). PHP’s magic_quotes_gpc should be turned OFF. It basically sets the magic quotes for GPC (Get/POST/Cookie) operations. When magic quotes are ON, all ‘ (single-quote), ” (double quote), \ (backslash) and NULL’s are escaped with a backslash automatically.

8). PHP’s safe_mode should be turned OFF. According to PHP documentation, “The PHP safe mode is an attempt to solve the shared-server security problem. It is architecturally incorrect to try to solve this problem at the PHP level, but since the alternatives at the web server and OS levels are not very realistic, many people use safe mode” [18].

9). PHP’s register_globals directive should also be turned OFF. If register_globals is turned ON, then one can inject malicious code using this directive which is a huge security concern.

10). PHP’s allow_url_fopen should be disabled. According to PHP documentation, “This option enables the URL—aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote

files using the ftp or http protocol, some extensions like zlib may register additional wrappers” [10, 18].

In order to securely test and develop a Joomla! website, following steps should be observed:

1). The most important rule is to setup a backup and recovery process. This is the best way to recover data in the event of hardware failure (such as power fail, dead hard drive, server theft), broken website due to wrong upgrade of either components, modules or plugins or an attacked website.

2). One should always develop and test on a local machine and then upload the changes on the live website.

3). Use of Integrated Development Environment (IDE) has many benefits such as quick navigation to a type; auto completion of names of all members when one cannot remember exactly; giving warning as we type, quickly adding notes so that others can hover over it; easy to run unit tests from the same window; integrate source control.

4). Using a sub versioning system (SVN) to be able to roll back to an earlier version [18].

Attacks on Joomla! Website

There are two major attacks that can take place on Joomla! website. They are:

- 1). SQL Injections.
- 2). Remote File Includes.

1). *SQL Injections*: The entire Joomla! website is managed through SQL database. They are the heart of the Joomla! website. It stores the content, user ID's and password which are the most precious resource of the entire Joomla! website. If someone gets access to this valuable asset, he can gather all the private information and can hack the entire website [21].

“SQL injection is a query put to an SQL database where data input was expected AND the application does not correctly filter the input” [21]. When a request to a page is made in Joomla!, it forms a query for the database. SQL database does not suspect of a malformed question and will do its function to process it. Older version of Joomla! had passwords stored as a Message-Digest algorithm 5 (MD5) hashes which could be decoded easily. Newer version of Joomla! has password calculated using “salt”, thus making it impossible to decode it. Salting is a method in cryptography to secure passwords [21].

There are few tests to check SQL injection vulnerabilities. In the login form, one can insert single quotes to test as follows: 1=1 - -

Also, one may try inputting the following [21]:

' OR 1=1 - -

" OR 1=1 - -

' OR 'x'='x

Methods to Prevent SQL Injections

a). Joomla! developer should *always* validate the user input, that is, test for type, format, length and range.

- b). File uploads should be restricted to the expected file types because a hacker may use this to upload malicious code.
- c). Always enforce the size as a hacker can embed a DROP TABLE statement in a text field or an INSERT statement. If the size is restricted then buffer overflow will be prevented.
- d). Content of the string variables should be tested and only expected values should be allowed. Common entries such as binary data, escape sequences, and comment characters should be rejected.
- e). SQL statements should not be allowed directly from the user input field.
- f). Concatenation of user input that is not validated should not be allowed since string concatenation is the primary point of entry for a script injection.
- g). Proper user privileges should be assigned. Database should not be connected as an Admin, Super-Admin or the Database Owner as they will be kept only for administrative purposes [21].

2). *Remote File Includes*: This is a technique to attack the website from a remote computer. This attack can be prevented by turning OFF PHP's Register_Globals. This ensures that the \$page variable is not treated as a global variable, thus does not allow an insertion. Also, allow_url_fopen should be turned OFF, which disables PHP file functions such as file_get_contents(). This function may retrieve the entire content of the website; thus allowing an attacker to break-in easily [21].

TABLE 2. Preconfigured .htaccess file shipped with Joomla! [19]

```

# @version $Id: htaccess.txt 10492 2008-07-02 06:38:28Z ircmaxell $
# @package Joomla
# @copyright Copyright (C) 2005—2008 Open Source Matters. All rights reserved.
# @license http://www.gnu.org/copyleft/gpl.html GNU/GPL
# Joomla! is Free Software

#####
# READ THIS COMPLETELY IF YOU CHOOSE TO USE THIS FILE
# The line just below this section: 'Options +FollowSymLinks' may cause problems with some server
configurations. It is required for use of mod_rewrite, but may already be set by your server administrator
in a way that disallows changing it in your .htaccess file. If using it causes your server to error out,
comment it out (add # to beginning of line), reload your site in your browser and test your sef url's. If they
work, it has been set by your server administrator and you do not need it set here.
## Can be commented out if causes errors, see notes above.
Options +FollowSymLinks

# mod_rewrite in use

RewriteEngine On

Begin – Rewrite rules to block out some common exploits. If you experience problems on your site block
out the operations listed below. This attempts to block the most common type of exploit attempts to
Joomla!
# Block out any script trying to set a mosConfig value through the URL
RewriteCond %{QUERY_STRING} mosConfig_[a-zA-Z_]{1,21}(=|%3D) [OR]
# Block out any script trying to base64_encode crap to send via URL
RewriteCond %{QUERY_STRING} base64_encode.*(.*?) [OR]
# Block out any script that includes a <script> tag in URL
RewriteCond %{QUERY_STRING} (<|%3C).*script.*(\\>|%3E) [NC,OR]
# Block out any script trying to set a PHP GLOBALS variable via URL
RewriteCond %{QUERY_STRING} GLOBALS(=|\\[\\%[0-9A-Z]{0,2}) [OR]
# Block out any script trying to modify a _REQUEST variable via URL
RewriteCond %{QUERY_STRING} _REQUEST(=|\\[\\%[0-9A-Z]{0,2})
# Send all blocked request to homepage with 403 Forbidden error!
RewriteRule ^(.*)$ index.php [F,L]
##### End—Rewrite rules to block out some common exploits
# Uncomment following line if your web server's URL is not directly related to physical file paths.
Update Your Joomla! Directory (just / for root)

# RewriteBase /

##### Begin – Joomla! core SEF Section
#
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_URI} !^/index.php
RewriteCond %{REQUEST_URI} (!\\\.php|\\.html|\\.htm|\\.feed|\\.pdf|\\.raw|/[^.]*)$ [NC]
RewriteRule (.*) index.php
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization},L]
#
##### End – Joomla! core SEF Section

```

CHAPTER 6

WEBSITE CONVERSION USING SCRAPER

The basic feature of a Scraper is to get the content from the HTML page into SQL database.

Database Architecture

Scraper affects only 4 tables of Joomla!'s MySQL database. The database tables used by the Scraper are:

- 1). **jos_content**: This table stores the main content (body) of the Joomla! website.

id is the primary field and acts as an auto-incrementing key for the table.

Table 3 shows the structure of **jos_content** with their type and default values.

- 2). **jos_menu**: This table stores the navigation of the Joomla! website.

id is the primary field and acts as an auto-incrementing key for the table.

Table 4 shows the structure of **jos_menu** with their type and default values.

- 3). **jos_sections**: This table stores the section information of the Joomla! website.

id is the primary field and acts as an auto-incrementing key for the table.

Table 5 shows the structure of **jos_sections** with their type and default values.

- 4). **jos_categories**: This table stores the category information of the Joomla!

website. *id* is the primary field and acts as an auto-incrementing key for the table.

Table 6 shows the structure of **jos_categories** with their type and default values.

TABLE 3. Database Structure of jos_content [15]

<u>Field</u>	<u>Type</u>	<u>Null</u>	<u>Default</u>
id	int(11)	No	
title	varchar(255)	No	
alias	varchar(255)	No	
title_alias	varchar(255)	No	
introtext	mediumtext	No	
fulltext	mediumtext	No	
state	tinyint(3)	No	0
sectionid	int(11)	No	0
mask	int(11)	No	0
catid	int(11)	No	0
created	datetime	No	0000-00-00 00:00:00
created_by	int(11)	No	0
created_by_alias	varchar(255)	No	
modified	datetime	No	0000-00-00 00:00:00
modified_by	int(11)	No	0
checked_out	int(11)	No	0
checked_out_time	datetime	No	0000-00-00 00:00:00
publish_up	datetime	No	0000-00-00 00:00:00
publish_down	datetime	No	0000-00-00 00:00:00
images	text	No	
urls	text	No	
attribs	text	No	
version	int(11)	No	1
parentid	int(11)	No	0
ordering	int(11)	No	0
metakey	text	No	
metadesc	text	No	
access	int(11)	No	0
hits	int(11)	No	0
metadata	text	No	

TABLE 4. Database Structure of jos_menu [15]

<u>Field</u>	<u>Type</u>	<u>Null</u>	<u>Default</u>
id	int(11)	No	
menutype	varchar(75)	Yes	<i>NULL</i>
name	varchar(255)	Yes	<i>NULL</i>
alias	varchar(255)	No	
link	text	Yes	<i>NULL</i>
type	varchar(50)	No	
published	tinyint(1)	No	0
parent	int(11)	No	0
componentid	int(11)	No	0
sublevel	int(11)	Yes	0
ordering	int(11)	Yes	0
checked_out	int(11)	No	0
checked_out_time	datetime	No	0000-00-00 00:00:00
pollid	int(11)	No	0
browserNav	tinyint(4)	Yes	0
access	tinyint(3)	No	0
utaccess	tinyint(3)	No	0
params	text	No	
lft	int(11)	No	0
rgt	int(11)	No	0
home	int(1)	No	0

Each Joomla! article is contained within a section and its respective category.

Scraper automatically assigns a particular article to its respective section and category from the navigation (breadcrumb) of the website.

TABLE 5. Database Structure of jos_sections [15]

<u>Field</u>	<u>Type</u>	<u>Null</u>	<u>Default</u>
id	int(11)	No	
title	varchar(255)	No	
name	varchar(255)	No	
alias	varchar(255)	No	
image	text	No	
scope	varchar(50)	No	
image_position	varchar(30)	No	
description	text	No	
published	tinyint(1)	No	0
checked_out	int(11)	No	0
checked_out_time	datetime	No	0000-00-00 00:00:00
ordering	int(11)	No	0
access	tinyint(3)	No	0
count	int(11)	No	0
params	text	No	

Basic Algorithm of a Scraper

The basic algorithm of the Scraper is as follows:

Step 1: Read the contents of the entire static HTML website by parsing each file and directory.

Step 2: Extract the required data from each static HTML web page and store it in the file in the form of SQL query.

Step 3: Run this query from the stored file on Joomla!'s database in order to get the entire website, which is dynamic and is now Joomla! administered.

TABLE 6. Database Structure of jos_categories [15]

Field	Type	Null	Default
id	int(11)	No	
parent_id	int(11)	No	0
title	varchar(255)	No	
name	varchar(255)	No	
alias	varchar(255)	No	
image	varchar(255)	No	
section	varchar(50)	No	
image_position	varchar(30)	No	
description	text	No	
published	tinyint(1)	No	0
checked_out	int(11)	No	0
checked_out_time	datetime	No	0000-00-00 00:00:00
editor	varchar(50)	Yes	NULL
ordering	int(11)	No	0
access	tinyint(3)	No	0
count	int(11)	No	0
params	text	No	

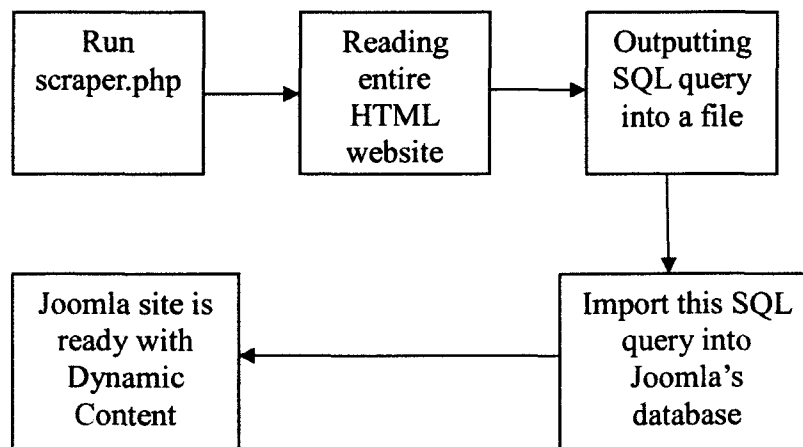


FIGURE 10. Pictorial representation of Scrapper Algorithm.

In order to implement the above algorithm, one requires the title of the web page, menu in which the item needs to be inserted and the main body of the web page. To serve these three purposes, three functions are designed as follows (see Appendix for source code).

Function Parsetitle (\$datafile)

The main purpose of this function is to get the title from the HTML page. In order to accomplish this, a regular expression is written to match the title tag in the HTML file. By using the `preg_match()` function of PHP, the title match for the HTML page is obtained that was parsed [10].

Description for `preg_match()` as in PHP Manual is as follows [10]:

int preg_match (string \$pattern, string \$subject [, array &\$matches])

where *pattern*: The pattern to search for, as a string (Regular expression)

subject: The input string

matches: If matches is provided, then it is filled with the results of search.

`$matches [0]` will contain the text that matched the full pattern, `$matches [1]` will have the text that matched the first captured parenthesized subpattern, and so on.

Function getbody (\$datafile)

The main purpose of this function is to get the body from the HTML page. By using the `preg_match()` function of PHP, the body match for the HTML page is obtained that was parsed [10].

Function getbreadcrumb (\$datafile)

The main purpose of this function is to get the Breadcrumbs (navigation part) from the HTML page. This is the most challenging part as here one has to do replacements according to the requirements of the website. By using the preg_match() function of PHP, the breadcrumb match for the HTML page is obtained that was parsed.

Function gethtml()

The main purpose of this function is to get the entire SQL query from the HTML page. The task can be sub-divided into many sub tasks.

First task will be reading the HTML files from the entire website by using Recursive Directory Iterator function of PHP. It traverses through all the directories and sub-directories of the entire website and reads the HTML files contained within the website [10].

For each directory, the function performs the following tasks:

- i). Gets the breadcrumbs path using GetBreadCrum() function,
- ii). Gets the title of the page using ParseTitle() function, Title contains the entire title with the path, so it is further trimmed and stored in the array.
- iii). Retrieve sections using breadcrumbs Array, Section ID, Category ID.
- iv). Similarly, categories are retrieved using breadcrumbs array, Section ID and Category ID.
- v). A Hash table is created to find out the respective child item of their respective parent.

Now, the existence of the item is checked in the hash table and if the item is found, then one has to grab the parent id of that particular item and then inserted into the hash table with the article index. If the item is not found, then the item has to be created in the hash table and a parent id needs to be assigned with that particular article index. Repeat step (v) until the complete hash table is ready.

Creating a Menu Structure

For each array keys from the hash table, the following step is executed.

Foreach (array_keys(\$menuhash) as \$key)

Using the explode function of PHP, the actual title of the page from the array key is extracted. Supposedly, if the title of the page is empty, then the existence of the title and zero string length is checked; if this is true then the unknown title to that page is assigned. After this, an insert query for the Menu is created; this will generate the menu items of the website.

This process is repeated until the whole menu is created. Further, the whole string containing the article id, parent id, title, main menu is appended in a file as an insert statement.

Similarly, the same process is repeated for sections and categories. Thus, one gets a huge file of SQL insert statements. This SQL file is imported directly into the Joomla!'s MySQL database. After the query is executed, all the menu items are generated with their respective parent and child items. Hence, the whole website is dynamically generated from the static HTML website.

TABLE 7. Comparison Chart between Manual Website Conversion and Implementation using Scraper [22]

<u>Manual Website Conversion</u>	<u>Implementation using Scraper—Joomla! Website</u>
Content is static HTML in web page form; it requires technical expertise and multiple instances have to be edited individually on each page.	Content is dynamic stored in MySQL database; it can be changed with no technical expertise and a single change will effect sitewide change.
Maintenance is time consuming and tedious.	Maintenance is fast and easy.
No PHP programming is used.	Scraper is implemented in PHP.
This involves client side programming.	PHP is an open-source server side scripting language.
Data, view and logic are all together in the website.	MVC structure is implemented which separates logic, data and view.
This does not use template file structure.	This uses template file structure.
This does not use components, modules and plugins.	This uses components, modules and plugins.
When HTTP request is made to the web server, the HTML content is delivered “as it is”.	When HTTP request is made to the web server, each page is built from the information stored in various files such as images, database and logic.
This might require special web authoring tools such as Dreamweaver or FTP access to the website.	Content can be revised from anywhere with internet connection and no special web authoring tools.
Less flexible.	More flexible.
High maintenance costs.	Low maintenance costs.
Breaking the visual style of the website is more often due to poor HTML coding.	Breaking the visual style of the website is almost impossible as it uses templates.
Data management is not very efficient as the website expands.	Data management is very efficient as the website expands.

CHAPTER 7

EXAMPLE OF SCRAPER IMPLEMENTATION

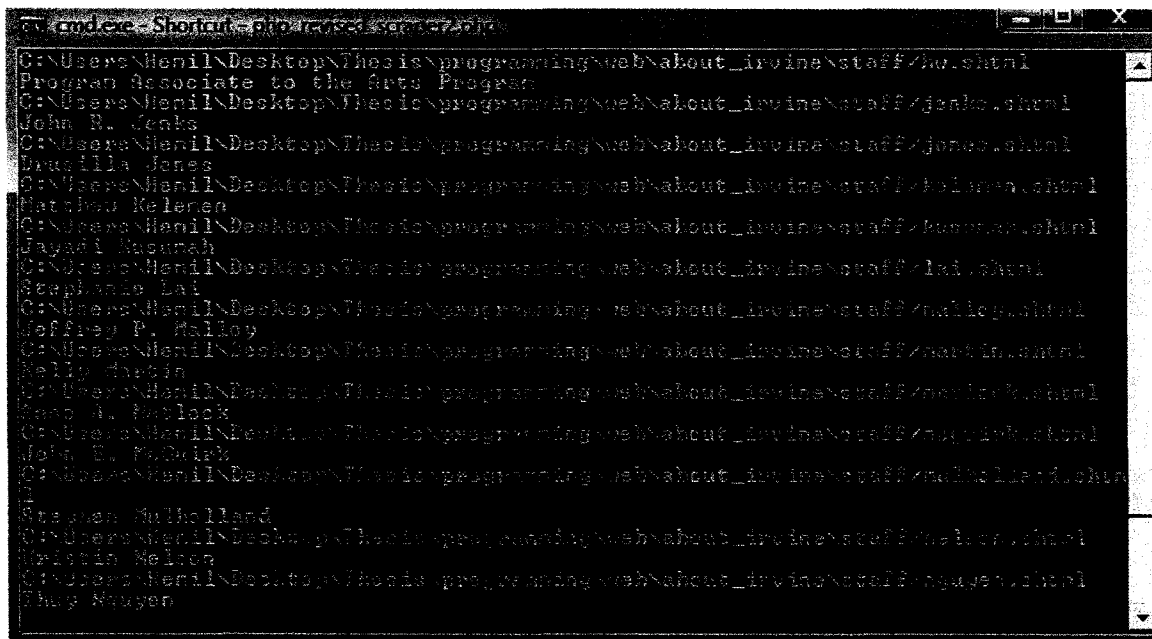
In this chapter, I will present the implementation of the Scraper and build a dynamic Joomla! website from static HTML website.

I have taken one sample HTML website with approximately 600 static pages. The Scraper program is then executed which would recursively parse within all the directories and sub-directories. The program will fetch the useful information needed to build Joomla!'s database. The information extracted is the page title; body and the navigation of the website that is stored in the form of SQL insert statements in a file. This SQL file is then imported into Joomla!'s database, which builds up the dynamic website.

Usefulness of the Scraper

Scraper is a very easy to use program for all operating systems and has a lot of efficacy. It is so straightforward to run and within minutes a complete dynamic content management website can be built from any static HTML website. One just requires this code and the entire website to convert the static HTML website to a Joomla! based website. In addition, it is such a handy program that the user only needs to change some Regular expressions as per the website's internal HTML code. With a Joomla! managed website, users can update, modify the website content easily and with minimum technical effort. They do not have to be anxious about the implementation of navigation as

Joomla! automatically manages it. One can easily alter the design of the entire website in minutes by just changing the Joomla! template. Joomla! is supported by several components, modules and plugins that can be effortlessly installed in no time. The content extraction from MySQL database is very fast in Joomla!; multiple requests to the same page by several users does not affect the load time of the website, since Joomla! uses caching, whereas the load time in static HTML website is higher. Hence, due to all these advantages, it is the time to switch to dynamic content management system like Joomla! Also our tool is fast; it took approximately 21 seconds to generate 2541 SQL statements out of 600 pages static HTML website pages. Figure 11 shows the Scraper code running in command prompt of windows.

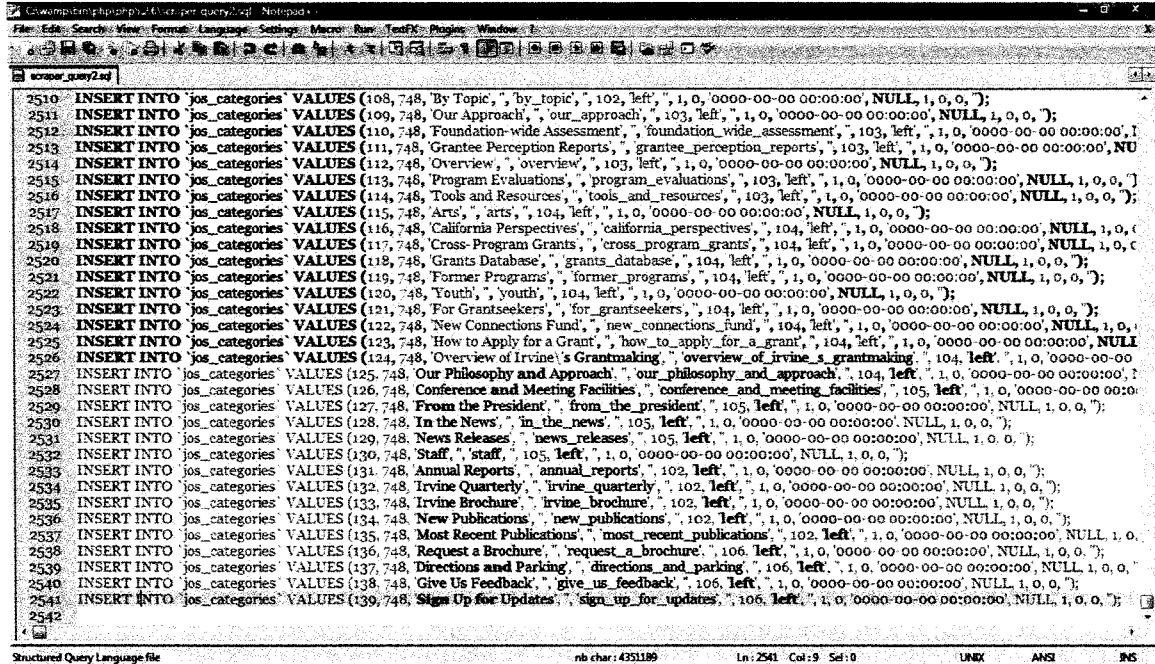


```
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\ho.shtml
Program Associate to the Arts Program
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\jenks.shtml
John R. Jenks
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\jones.shtml
Druzilla Jones
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\kolman.shtml
Matthew Kelenen
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\kumar.shtml
Jayadi Kumarah
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\lai.shtml
Stephanie Lai
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\halley.shtml
Jeffrey P. Halley
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\martin.shtml
Kelly Martin
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\mcclock.shtml
Eugene A. McClock
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\mcquinn.shtml
John E. McQuinn
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\mullholland.shtml
Stephen Mullholland
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\nelson.shtml
Justin Nelson
C:\Users\Nenil\Desktop\Thesis\programming\web\about_irvine\staff\nguyen.shtml
Thuy Nguyen
```

FIGURE 11. Screenshot of a Scraper program.

Figure 12 shows the SQL file generated as a result of running the Scraper code.

This file is then imported into Joomla!'s database to generate a dynamic content management website.



```
2510 INSERT INTO jos_categories VALUES (108,748,'By Topic','by_topic','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2511 INSERT INTO jos_categories VALUES (109,748,'Our Approach','our_approach','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2512 INSERT INTO jos_categories VALUES (110,748,'Foundation-wide Assessment','foundation_wide_assessment','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2513 INSERT INTO jos_categories VALUES (111,748,'Grantee Perception Reports','grantee_perception_reports','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2514 INSERT INTO jos_categories VALUES (112,748,'Overview','overview','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2515 INSERT INTO jos_categories VALUES (113,748,'Program Evaluations','program_evaluations','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2516 INSERT INTO jos_categories VALUES (114,748,'Tools and Resources','tools_and_resources','103,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2517 INSERT INTO jos_categories VALUES (115,748,'Arts','arts','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2518 INSERT INTO jos_categories VALUES (116,748,'California Perspectives','california_perspectives','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2519 INSERT INTO jos_categories VALUES (117,748,'Cross-Program Grants','cross_program_grants','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2520 INSERT INTO jos_categories VALUES (118,748,'Grants Database','grants_database','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2521 INSERT INTO jos_categories VALUES (119,748,'Former Programs','former_programs','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2522 INSERT INTO jos_categories VALUES (120,748,'Youth','youth','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2523 INSERT INTO jos_categories VALUES (121,748,'For Grantseekers','for_grantseekers','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2524 INSERT INTO jos_categories VALUES (122,748,'New Connections Fund','new_connections_fund','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2525 INSERT INTO jos_categories VALUES (123,748,'How to Apply for a Grant','how_to_apply_for_a_grant','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2526 INSERT INTO jos_categories VALUES (124,748,'Overview of Irvine's Grantmaking','overview_of_irvine_s_grantmaking','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2527 INSERT INTO jos_categories VALUES (125,748,'Our Philosophy and Approach','our_philosophy_and_approach','104,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2528 INSERT INTO jos_categories VALUES (126,748,'Conference and Meeting Facilities','conference_and_meeting_facilities','105,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2529 INSERT INTO jos_categories VALUES (127,748,'From the President','from_the_president','105,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2530 INSERT INTO jos_categories VALUES (128,748,'In the News','in_the_news','105,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2531 INSERT INTO jos_categories VALUES (129,748,'News Releases','news_releases','105,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2532 INSERT INTO jos_categories VALUES (130,748,'Staff','staff','105,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2533 INSERT INTO jos_categories VALUES (131,748,'Annual Reports','annual_reports','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2534 INSERT INTO jos_categories VALUES (132,748,'Irvine Quarterly','irvine_quarterly','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2535 INSERT INTO jos_categories VALUES (133,748,'Irvine Brochure','irvine_brochure','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2536 INSERT INTO jos_categories VALUES (134,748,'New Publications','new_publications','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2537 INSERT INTO jos_categories VALUES (135,748,'Most Recent Publications','most_recent_publications','102,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2538 INSERT INTO jos_categories VALUES (136,748,'Request a Brochure','request_a_brochure','106,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2539 INSERT INTO jos_categories VALUES (137,748,'Directions and Parking','directions_and_parking','106,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2540 INSERT INTO jos_categories VALUES (138,748,'Give Us Feedback','give_us_feedback','106,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2541 INSERT INTO jos_categories VALUES (139,748,'Sign Up for Updates','sign_up_for_updates','106,left','1,0','0000-00-00 00:00:00',NULL,1,0,0,);
2542
```

FIGURE 12. Screenshot of the SQL file (2541 insert statements) generated by the Scraper.

Figure 13 shows one static HTML page from the example website. Here the content is loaded from a static HTML file residing on the server. The top navigation is a table which is loaded from the other HTML page. Also, the left navigation is a JavaScript code that needs to be manually updated whenever a navigation link has been changed. The website search is manually implemented as and when there is an update to any web page, it needs to be re-indexed.

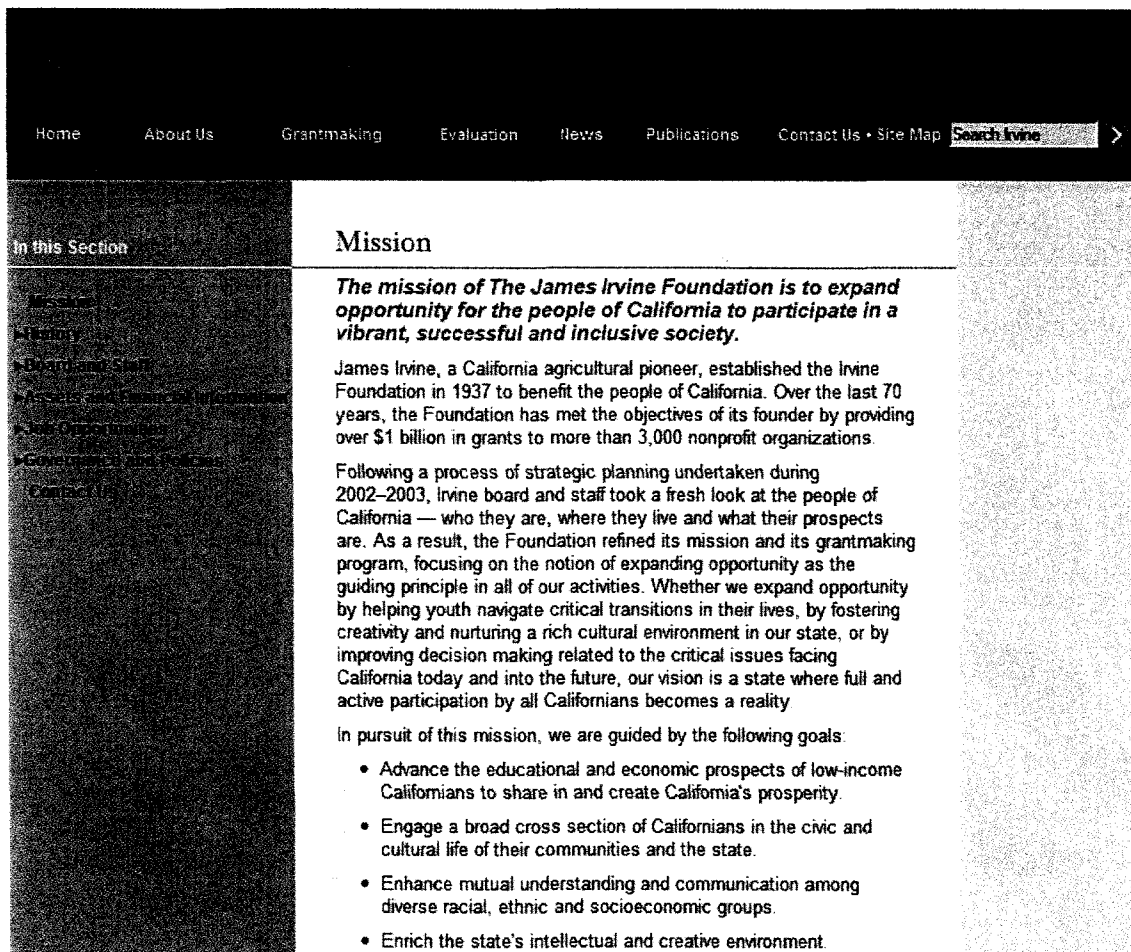


FIGURE 13. Screenshot of a sample static HTML page.

Figure 14 shows the same page generated by Joomla! Here, the content is loaded from the MySQL database. Navigation is built-in Joomla! and can be changed using the Joomla! administrative screen. Search is built-in Joomla! and is dynamic. Whenever the content is changed in a web page, the search feature is automatically updated with the new content and it is available in the search engine immediately.

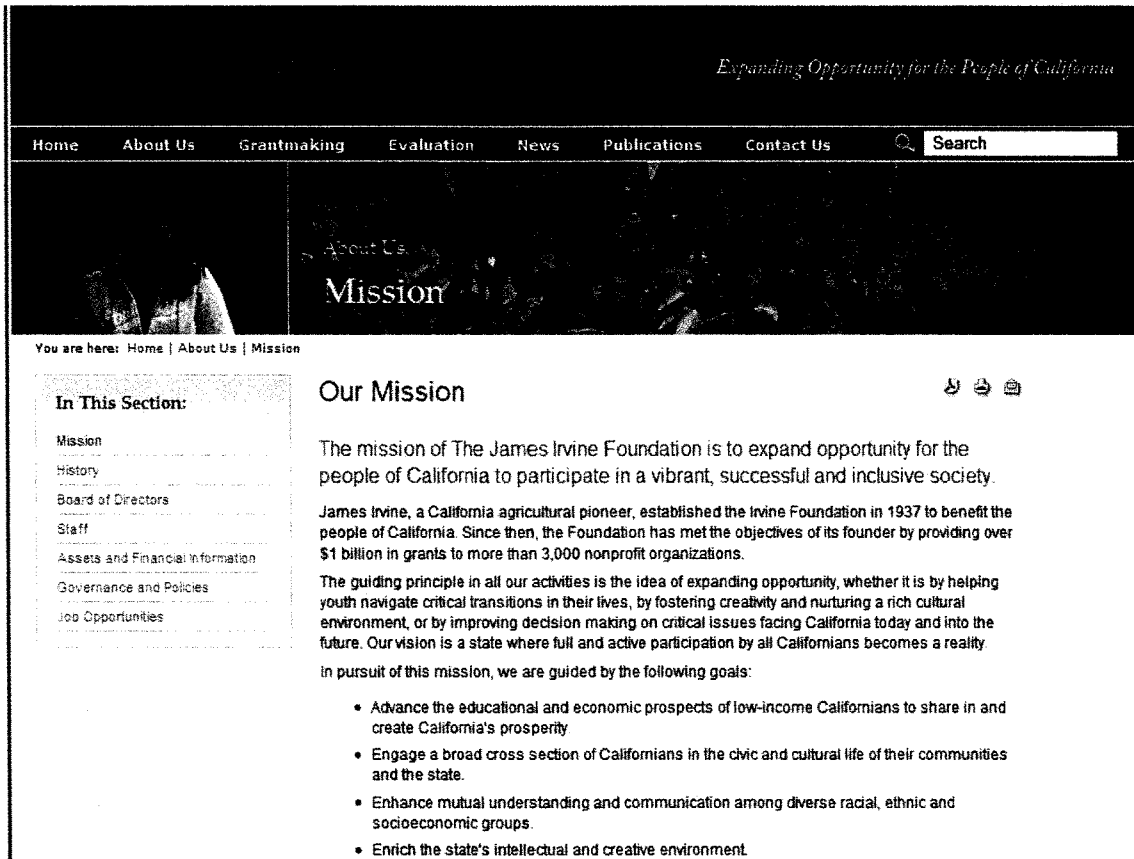


FIGURE 14. Screenshot of a sample page in Joomla! generated by the Scraper.

Future developments to the Scraper (Wish List)

1). *GUI-Interface*: The graphical user interface (GUI) can be improved by adding the feature that allows a user to add any local or remote HTML website address which needs to be converted to a Joomla! site. Also, the three elements (Section, Category and Articles) which are mainly required by Joomla! can be added as a interface that allows a user to select the Regular expressions from the pre-loaded basic expressions in the drop-down menu. This helps the user to quickly choose it from the list of available expressions as per the website's code.

2). *Complete Automation*: Instead of storing the SQL statements in a external file, a SQL connection is made to the Joomla!'s database directly through the Scraper that does not involve an extra last step to convert the website. This will update Joomla!'s MySQL database at the end of this program.

3). *Implementation of Scraper as Joomla!'s component*: If the Scraper is available to users as a Joomla! Component, then it would be more productive to convert any static HTML website by just installing this Component. No installation of PHP will be needed.

APPENDIX
SCRAPER CODE

PHP CODE OF SCRAPER

```
<?php

// Function to get Title the whole website
function ParseTitle($datafile)
{
    //echo $datafile;
    //Opening a file and storing it in string data
    $fh = fopen("$datafile", "rb");
    $data = fread($fh, filesize($datafile));

    // Regex to match the title tag in HTML page—This can be changed as per the requirements of the website
    if (preg_match("/.* Insert Regex to match the title tag */", $data, $matches))
    {
        //Match Found
    } else
    {
        //echo "\nA Title match was not found. $datafile\n";
        //fgets(STDIN);
    }
    fclose($fh); // Closing the file
}

// Function to get Body from the HTML website
function GetBody($datafile)
{
    // Regex to match the <td class="centercol"> tag in HTML pages—This can be changed as per the
    requirements of the website

    $fh = fopen("$datafile", "rb");
    $data = fread($fh, filesize($datafile));
    // This can be changed as per the requirements of the website
    if (preg_match("/.* Insert Regex to match the body tag */", $data, $matches))
    {
        //Match Found
    } else
    {
        //echo "\nA Body match was not found. $datafile\n";
    }
    fclose($fh);
}

function GetBreadCrum($datafile)
{
    // Regex to match the breadcrumbs in HTML pages
    $fh = fopen("$datafile", "rb");
    $data = fread($fh, filesize($datafile));
    //echo $datafile;
    //This can be changed as per the requirements of the website
```



```

if (preg_match('/* Insert Regex to match the breadcrumb tag */', $data, $strailmatches))
    {
        //Match Found
    } else
    {
        //echo "\nA Bread Crum match was not found. $datafile\n";
        //fgets(STDIN);
    }
// This can be changed as per the requirements of the website

fclose($fh);
}

function GetHtml()
{
    $path = realpath('C:\Users\Hemil\Desktop\Thesis\programming\web');
    $dir = new RecursiveDirectoryIterator($path);
    $newfile = 'scraper_query.sql';
    $resource_handle = fopen($newfile, 'ab');
    $menuind = 100;
    $artindex = 100;
    $catid = 100;
    $sectid = 100;
    $menuhash = array();
    $sectarray = array();
    $catarray = array();

/* DELETE Query to delete all the existing sections, categories, menu and articles

    file_put_contents($newfile, "DELETE FROM `jos_sections` WHERE id>=100;\n",
FILE_APPEND);
    file_put_contents($newfile, "DELETE FROM `jos_categories` WHERE id>=100;\n",
FILE_APPEND);
    file_put_contents($newfile, "DELETE FROM `jos_menu` WHERE menutype='mainmenu';\n",
FILE_APPEND);
    file_put_contents($newfile, "DELETE FROM `jos_content` WHERE created='2008-07-31
00:00:00';\n", FILE_APPEND);

foreach (new RecursiveIteratorIterator($dir) as $file)
{
    if (preg_match("/.shtml$/", $file->getFilename()))
    {
        $datafile = $file->getPath() . "/" . $file->getFilename();
        $crumbs = GetBreadCrum($datafile);
        $title = ParseTitle($datafile);
        $titleparts = explode(":", $title);
        $title = trim(array_pop($titleparts));
        $title = str_replace("'", "\'", $title);
        $titlealias = strtolower($title);
        $titlealias = ereg_replace('[^A-Za-z0-9]', '_', $titlealias);
        if (!$title || !strlen($title))
            $title = 'unknowntitle' . $artindex;

```

```

//print_r ("\n$title\n");

// Retrieve Sections
$mysectid = 0;
$mycatid = 0;
if (!$sectarray[$scrums[0]])
{
    $sectarray[$scrums[0]] = $sectid;
    $mysectid = $sectid;
    $sectid++;
} else {
    $mysectid = $sectarray[$scrums[0]];
}

// Retrieve Categories
if (!$catarray[$scrums[1]])
{
    $catarray[$scrums[1]] = array($catid, $mysectid);
    $mycatid = $catid;
    $catid++;
} else {
    $mycatid = $catarray[$scrums[1]][0];
}

$body = GetBody($datafile); // replace ' with \
$body = str_replace("'", "\'", $body);
$body = str_replace("\n", "\\n", $body);
$body = str_replace("\r", "\\r", $body);

// creating sql insert for Content—Body
$sql = "INSERT INTO `jos_content` VALUES ($artindex, '$title', '$titlealias', ",
'$body', ", '1', '$mysectid', '0', '$mycatid', '2008-07-31 00:00:00', '62', ", '2008-07-31 00:00:00', '62', '0',
'0000-00-00 00:00:00', '2008-01-26 11:30:00', '0000-00-00 00:00:00', ", ",
'show_title=\nlink_titles=\nshow_intro=\nshow_section=\nlink_section=\nshow_category=\nlink_categ
ory=\nshow_vote=\nshow_author=\nshow_create_date=\nshow_modify_date=\nshow_pdf_icon=\nsho
w_print_icon=\nshow_email_icon=\nlanguage=\nkeyref=\nreadmore=', '6', '0', '7', ", ", '0', '39',
'robots=\nauthor=');\n";

$parentid = 0;
//print_r($scrums);

// Creating a Hash table
for ($i = 0; $i < count($scrums); $i++)
{
    $hashkey = "";
    for ($t = 0; $t <= $i; $t++)
    {
        $hashkey .= "/" . $scrums[$t];
    }
}

```

```

// check to see if the item exists
if (isset($menuhash[$hashkey]) && $menuitem = $menuhash[$hashkey])
{
    // yes, grab the parent id
    $parentid = $menuitem["id"];
    if ($i == count($crumbs) - 1)
    {
        $menuitem["aid"] = $artindex;
    }
    $menuhash[$hashkey] = $menuitem;
} else
{
    // no, create it
    $menuitem = array();
    $menuitem["id"] = $menuind;
    $menuitem["pid"] = $parentid;
    $menuitem["aid"] = $artindex;
    if ($i == count($crumbs) - 1)
    {
        $menuitem["aid"] = $artindex;
    }
    $parentid = $menuind;
    $menuind++;
    //print("\nhashkey: " . $hashkey);
    $menuhash[$hashkey] = $menuitem;
    //print ("\nmenuitem: " . $menuhash[$hashkey] );
}
}

// add main menu item

file_put_contents($newfile, $insql, FILE_APPEND);
$artindex++;
}

$mysectid=-1;
// Retrieve Sections
if (!($sectarray[$crumbs[0]]))
{
    $sectarray[$crumbs[0]] = $sectid;
    $mysectid = $sectid;
    $sectid++;
} else {
    $mysectid = $sectarray[$crumbs[0]];
}

// Retrieve Categories
if (!($catarray[$crumbs[1]]))
{
    $catarray[$crumbs[1]] = array($catid, $mysectid);
    $catid++;
}
}

```

```

        fclose($resource_handle);

//print_r($menuhash);
foreach (array_keys($menuhash) as $key)
{
    $title = array_pop(explode('/', $key));
    $titlealias = strtolower($title);
    $titlealias = ereg_replace('[^A-Za-z0-9]', '_', $titlealias);
    $menuitem = $menuhash[$key];
    $aid = $menuitem["aid"];
    $pid = $menuitem["pid"];
    $sid = $menuitem["id"];
    $title = str_replace("''", "\'", $title);
    if (!$title || !strlen($title))
        $title = 'unknowntitle' . $sid;

    //print_r ("$title\n");

// creating sql insert for Menus
    $insql = "INSERT INTO `jos_menu` VALUES ($sid, 'mainmenu', '$title', '$titlealias',
'index.php?option=com_content&view=article&id=$aid', 'component', '1', '$pid', '20', '0', '9', '0', '0000-00-
00 00:00:00', 0, '0', '0', '0',
'show_noauth=\nshow_title=\nlink_titles=\nshow_intro=\nshow_section=\nlink_section=\nshow_categ
ory=\nlink_category=\nshow_author=\nshow_create_date=\nshow_modify_date=\nshow_item_navigati
on=\nshow_readmore=\nshow_vote=\nshow_icons=\nshow_pdf_icon=\nshow_print_icon=\nshow_em
ail_icon=\nshow_hits=\nfeed_summary=\npage_title=\nshow_page_title=1\npageclass_sfx=\nmenu_i
mage=-1\nsecure=0\n\n', '0', '0', '0');\n";

        //print($insql);
        file_put_contents($newfile, $insql, FILE_APPEND);
        file_put_contents($newfile, "# $key\n", FILE_APPEND);
}

// insert the home link
$mymenuind = $menuind+1;
$insql = "INSERT INTO `jos_menu` VALUES ($mymenuind, 'mainmenu', 'Home', 'home',
'index.php?option=com_content&view=frontpage', 'component', '1', '0', '20', '0', '1', '0', '0000-00-00
00:00:00', 0, '0', '0', '0',
'num_leading_articles=1\nnum_intro_articles=4\nnum_columns=2\nnum_links=4\norderby_pri=\norder
by_sec=front\nshow_pagination=2\nshow_pagination_results=1\nshow_feed_link=1\nshow_noauth=\n
show_title=\nlink_titles=\nshow_intro=\nshow_section=\nlink_section=\nshow_category=\nlink_categ
ory=\nshow_author=\nshow_create_date=\nshow_modify_date=\nshow_item_navigation=\nshow_read
more=\nshow_vote=\nshow_icons=\nshow_pdf_icon=\nshow_print_icon=\nshow_email_icon=\nshow
_hits=\nfeed_summary=\npage_title=\nshow_page_title=1\npageclass_sfx=\nmenu_image=-
1\nsecure=0\n\n', '0', '0', '0');\n";
    //print($insql);
    file_put_contents($newfile, $insql, FILE_APPEND);

    foreach (array_keys($sectarray) as $key){
        $title = $key;
        $titlealias = strtolower($title);

```

```

        $titlealias = ereg_replace('[^A-Za-z0-9]', '_', $titlealias);
        if (!$title || !strlen($title))
            $title = 'unknowntitle' . $sectid;
        $sectid = $sectarray[$key];
// creating sql insert for Sections
        $sinsql = "INSERT INTO `jos_sections` VALUES ($sectid, '$title', '', '$titlealias', '', 'content', 'left',
", 1, 0, '0000-00-00 00:00:00', 1, 0, 6, '');\n";

        file_put_contents($newfile, $sinsql, FILE_APPEND);
    }
// print_r($scatarray);
    foreach (array_keys($scatarray) as $key){
        $scat = $scatarray[$key];
        $sectid = $scat[1];
        $catid = $scat[0];
        $title = $key;
        $titlealias = strtolower($title);
        $titlealias = ereg_replace('[^A-Za-z0-9]', '_', $titlealias);
        if (!$title || !strlen($title))
            $title = 'unknowntitle' . $catid;
        $title = str_replace("'", "\'", $title);
        //$catid = $scatarray[$key];
        // creating sql insert for Categories
        $sinsql = "INSERT INTO `jos_categories` VALUES ($catid, $pid, '$title', '', '$titlealias', '',
$sectid, 'left', '', 1, 0, '0000-00-00 00:00:00', NULL, 1, 0, 0, '');\n";

        file_put_contents($newfile, $sinsql, FILE_APPEND);
    }
//echo "\nTotal Number of Html Files are :";
    //echo $scout;
} // End GetHtml()

// Main Part of Function Call

GetHtml();

?>

```

REFERENCES

REFERENCES

- [1]. INTERNET USAGE STATISTICS, Retrieved on October 7, 2009, from <http://www.internetworldstats.com/stats.htm>
- [2]. Dan Rahmel, "Beginning Joomla!," Apress, 2009.
- [3]. Knowing When You Need a CMS, Retrieved on June 1, 2009, from <http://www.techsoup.org/learningcenter/webbuilding/archives/page9378.cfm>
- [4]. CMS Website Design, Retrieved on June 15, 2009, from <http://www.navegabem.com/cms-website-design.html>
- [5]. The GNU General Public License - GNU Project - Free Software Foundation (FSF), Retrieved on June 17, 2009, from <http://www.gnu.org/licenses/gpl.html>
- [6]. Joomla!® The most powerful Open Source CMS, Retrieved on June 15, 2009, from <http://www.navegabem.com/cms-website-advantages.html>
- [7]. What is Joomla!?, Retrieved on June 15, 2009, from <http://www.joomla.org/about-joomla.html>
- [8]. Web Extractor, Web Scraper, Web Grabber, Web Scraping and Data Extraction Software, Retrieved on October 13, 2009, from <http://www.newprosoft.com/web-content-extractor.htm>
- [9]. Vote for Joomla!, Retrieved on June 15, 2009, from <http://www.joomla.org/announcements/general-news/5248-vote-for-joomla.html>
- [10]. PHP: What is PHP?, Retrieved on June 15, 2009, from <http://www.php.net/manual/en/intro-what-is.php>
- [11]. PHP: What can PHP do?, Retrieved on June 15, 2009, from <http://www.php.net/manual/en/intro-whatcando.php>
- [12]. MySQL Documentation, Retrieved on June 25, 2009, from <http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html>

- [13]. phpMyAdmin Documentation, Retrieved on June 20, 2009, from http://www.phpmyadmin.net/home_page/index.php
- [14]. Document Object Model (DOM), Retrieved on June 15, 2009, from <http://www.w3.org/DOM/>
- [15]. Joomla! Framework, Retrieved on June 1, 2009, from <http://docs.joomla.org/Framework>
- [16]. Developing a Model-View-Controller Component-Joomla! Documentation, Retrieved on June 1, 2009, from http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_1
- [17]. Joomla! MVC, Retrieved on June 30, 2009, from <http://www.phpeveryday.com/articles/Joomla-MVC-Creating-Controller-Section-P130.html>
- [18]. Joomla! Official Documentation, Retrieved on September 18, 2009, from <http://docs.joomla.org/>
- [19]. Preconfigured .htaccess file, Retrieved on Oct 1, 2009, from http://docs.joomla.org/Preconfigured_.htaccess
- [20]. Apache Core features, Retrieved on Oct 1, 2009, from <http://httpd.apache.org/docs/2.0/mod/core.html>
- [21]. Tom Canavan, "Joomla! Web Security: Secure your Joomla! Website from Common Security Threats with this easy-to-use Guide", Packt Publishing, 2008.
- [22]. Barrie M. North, "Joomla! A User's Guide: Building a Successful Joomla! Powered Website", Prentice Hall, 2007

Note: Joomla! documentation is licensed with the Joomla! Electronic Documentation license. <http://www.opensource-matters.org/index.php?Itemid=163>