

**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**WEB BASED COMPARE DIFFERENT DATED  
STOCKS SYSTEMS**

**Graduating Project  
COM- 400**

**Student: Mahsum Akbaş**

**Supervisor: Umit Soyer**

**Nicosia-2007**

## ACKNOWLEDGMENT

My gratitude for the following people cannot be expressed in this short acknowledgment

Firstly, I wish to thank who made this Project possible to me, my supervisor Mr. Umit Soyer.

I would like to my special thanks to my friends who share their knowledge with me, Murat Silinir and Fırat Silinir.

Also i would like to thanks to my family everytime be with me and my home friends because during three months, they helped me about everything and about idea of projects, Yaser Sevim and Hasan Öztürk

## ABSTRACT

Stock control is a system for keeping adequate control of stock levels and for preventing stock losses. Stock control, otherwise known as inventory control, is about how much stock you have at any one time, and how you keep track of it. It applies to every item you use to produce a product or service, from raw materials to finished goods. It covers stock at every stage of the production process, from purchase and delivery to using the stock and re-ordering. Efficient stock control will mean you have the right amount of stock in the right place at the right time. It ensures that capital is not tied up unnecessarily, and protects production when there are problems with the supply chain. This guide explains different stock control methods, will help you set one up and tells you where to find more information.

Everything you use to make your products, provide your services and to run your business is part of your stock. You can categorise stock further, according to its value. For example, you could put items into low, medium and high value categories. If you feel your stock levels are limited by capital, this will help you to plan expenditure on new and replacement stock. You may choose to concentrate resources on the areas of greatest value. However, low-cost items can be crucial to your production process and should not be overlooked. Deciding how much stock to keep depends on the size and nature of your business, and the type of stock involved. This might suit your business if it's in a fast-moving environment where products develop rapidly, the stock is expensive to buy and store, the items are perishable or replenishing stock is quick and easy. This might suit your business if sales are difficult to predict (and it is hard to pin down how much stock you need and when), you can store plenty of stock cheaply, the components or materials you buy are unlikely to go through rapid developments or they take a long time to re-order.

In sum, Project aims to constitute interactiveness, understandable, easiness, easy connection, good governace, good controlling and efficiency by the web based systems.

## INTRODUCTION

### TABLE OF CONTENTS

A web page is location on a computer network that makes information in the form of page or documents available to visitors who connect to the page by using a Web browser

A web browser can be publishes in the form of HTML pages, or in other document formats. To view the information available on a Web site, visitors use Web browser software programs ,like Internet Explorer , Netscape or Firefox, which translate HTML pages on Web site to text and graphics on their monitors.

A home page is an entry page for a set of web pages. It is a document written in HTML format that might be describe the content available on the site

The primary function of a home page is to introduce visitors to our Web site and help them navigate through its page

Chapter's gives techniques about HTML documents, Javascript, Apache Web Server, PHP and MySQL

**Chapter 1 :** Shows the basic HTML document and usage of basic HTML tags and its attributes.

**Chapter 2 :** Describe the Javascript fundamentals and usage with html forms.

**Chapter 3 :** Describe apache Web Server and how configuring

**Chapter 4 :** Explain the PHP basics and objects.

**Chapter 5 :** Explain information about the MySQL and shows the connection with database

**Chapter 6 :** Gives information about designing and used funtions in proect.

**Chapter 7 :** Applications codes of project

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENT</b>	i
<b>ABSTRACT</b>	ii
<b>INTRODUCTION</b>	iii
<b>TABLE OF CONTENTS</b>	iv
<b>CHAPTER ONE : HYPERTEXT MARKUP LANGUAGE - HTML</b>	<b>1</b>
1.1 What is HTML	1
1.2 What is an HTML File	1
1.2.1 HTM or HTML Extension	1
1.2.2 Headings	2
1.2.3 Paragraphs	2
1.2.4 Line Breaks	2
1.2.5 Comments in HTML	2
1.3 HTML Elements	2
1.3.1 Why Do We Use Lowercase Tags	3
1.4 Tag Attributes	3



1.4.1 Quote Styles , “red” or ‘red’ ?	3
<b>CHAPTER TWO : JAVASCRIPT</b>	<b>4</b>
2.1 Introduction	4
2.2 Declaring JavaScript	4
2.2.1 Alerts	5
2.2.2 Variables	6
2.2.3 Getting Information From User	6
2.2.4 Prompt	6
2.2.5 Document.WriteLine	7
2.3 Remote JavaScript	7
2.4 Link Events	8
2.4.1 onClick	8
2.4.2 onMouseOver and onMouseOut	9
2.5 Rollover Images	9
2.6 Status Bar	10
<b>CHAPTER THREE : APACHE WEB SERVER</b>	<b>11</b>
3.1 The WWW	11
3.1.1 NCSA	11
3.2 Introduction : What is Apache	11

3.1.1 The Apache Server	11
3.1.2 Apache's Architectures	12
3.1.3 More Recent History	12
3.1.4 The Future of Apache	12
3.1. 5 Support for Apache	13
3.3 Obtaining and Installing Apache	13
3.3.1 Hardware / Software Requirements	13
3.3.2 Advanced Installation	14
3.3.3 Editing The Configuration Script	14
3.3.4 Dynamic Shared Objects (DSO's)	15
3.4 Configuring	16
3.4.1 Configuration Files	16
3.4.2 Comanche	17
3.4.3 Starting , Stoping , Restarting	17
3.5 Integrating Apache With The Rest of Your Business	18

3.5.1 CGI	18
3.5.2 MIME Headers	18
3.5.3 Reading Client Input	19
3.5.4 mod_perl	19
3.5.5 SSL and E-commerce	20
3.5.6 Authentication	20
3.5.7 mod_auth	20
3.5.8 Log Files	22
3.6 A Conclusion for Apache	23
 <b>CHAPTER FOUR : PERSONAL HOME PAGE – PHP</b>	 <b>24</b>
4.1 What is PHP	24
4.2 Writing PHP	24
4.2.1 Basic PHP Syntax	24
4.2.2 Comments in PHP	25
4.2.3 Declaring PHP	25
4.3 Variables	26
4.3.1 Outputting Variables	27
4.3.2 Formatting Text	27
4.4 PHP Operators	28



4.4.1 Arithmetic Operators	28
4.4.2 Assignment Operators	29
4.4.3 Comparison Operators	29
4.4.4 Logical Operators	29
4.5 Conditional Statements	29
4.5.1 If...Else Statement	30
4.5.2 The Else if Statement	30
4.6 PHP String Processing	31
4.6.1 PHP String Functions	31
4.7 The Mail() Function	34
4.7.1 PHP Simple Text E-mail	34
4.7.2 PHP Mail Form	35
4.7.3 Requirements	36
4.7.4 Runtime Configurations	36
4.7.5 PHP Mail Functions	36
4.8 Cookies	37
4.8.1 How to Create a Cookie	37
4.8.2 How To Retrieve a Cookie Value	37
 <b>CHAPTER FIVE : MySQL</b>	 <b>38</b>
5.1 What is MySQL	38
5.2 Database Construction	38

5.2.1 Database and Login	38
5.3 Creating a Table	39
5.3.1 Fields	39
5.3.2 Creating a Table With PHP	39
5.3.3 The Contacts Database	40
5.4 Connect to MySQL Server	40
5.4.1 Using The Error Control Operator	41
5.4.2 Using The Die Function	41
5.5 Creating a Database	41
5.6 Primary Key and Auto Increment	43
5.7 PHP MySQL Function	43
5.7.1 Selecting a Database	45
5.7.2 Handling Errors	45
5.8 Inserting Data From a Form to a Database	46
5.9 ODBC With PHP	47
5.9.1 Connecting to an ODBC	47
5.9.2 Retrieving Records	48
5.9.3 Closing an ODBC Connection	48

<b>CHAPTER SIX : PROJECT DESCRIPTION</b>	<b>50</b>
6.1 Introduction	50
6.2 Customer Interface	50
6.3 Branch Interface	53
6.3 Branch Processes	56
6.4 Administrative Center Interface	59
 <b>CHAPTER SEVEN : PROJECT APPLICATION CODES</b>	 <b>62</b>
7.1 Customer Interface Codes	62
7.1.1 Index.php	62
7.1.2 mahsum_db.php	66
7.1.3 transaction.php	66
7.1.4 customer_function.php	67
7.2 Branch Interface Codes	69
7.2.1 Index.php	69
7.2.2 add_stock.php	71
7.2.3 edit_stock.php	74
7.2.4 Admin_manage.php	78
7.3 Administrative Center Interface Codes	81
7.3.1 Index.php	81

## CONCLUSION

## REFERENCES

87  
88

## 1.2 What Is HTML?

HTML (Hypertext Markup Language) is a markup language used to create web pages. It is a text-based language that uses tags to format text and create links. The basic structure of an HTML document is as follows:

```

<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>

```

The `<html>` tag is the root of the document. It contains two main sections: the `<head>` section, which contains meta-information about the document, and the `<body>` section, which contains the main content of the page. The `<title>` tag is used to specify the title of the page, and the `<h1>` tag is used to specify the main heading. The `</html>` tag is the closing tag for the root element.

## 1.3 Basic HTML Tags

HTML tags are used to format text and create links. The most common tags are:

- `<h1>` to `<h6>`: Heading tags, used to specify the main heading of the page.
- `<p>`: Paragraph tag, used to specify a paragraph of text.
- `<a href="url">`: Anchor tag, used to create a link to another page.
- ``: Image tag, used to insert an image into the page.
- `<pre>`: Preformatted text tag, used to display text as-is.
- `<code>`: Code tag, used to display text as code.
- `<br>`: Line break tag, used to insert a new line.
- `<hr>`: Horizontal rule tag, used to insert a horizontal line.

## 1.4 Creating a Simple Web Page

To create a simple web page, you need to create a text file with the following content:

```

<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>

```



# CHAPTER 1 – HYPERTEXT MARKUP LANGUAGE (HTML)

## 1.1 What is HTML

HTML (Hypertext Markup Language) the authoring language used to create documents on the World Wide Web. HTML defines the structure and layout of a Web document by using a variety of tags and attributes. All the information you'd like to include in your Web page fits in between the tags, and is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page. The markup tells the Web browser how to display a Web page's words and images for the user. Each individual markup code is referred to as an element. Some elements come in pairs that indicate when some display effect is to begin and when it is to end. HTML is a formal Recommendation by the World Wide Web Consortium (W3C) and is generally adhered to by the major browsers.

## 1.2 What is an HTML File?

HTML stands for Hyper Text Markup Language

An HTML file is a text file containing small markup tags

The markup tags tell the Web browser how to display the page

An HTML file must have an htm or html file extension

An HTML file can be created using a simple text editor

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

The first tag in your HTML document is <html>. This tag tells your browser that this is the start of an HTML document. The last tag in your document is </html>. This tag tells your browser that this is the end of the HTML document.

The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window.

The text between the <title> tags is the title of your document. The title is displayed in your browser's caption.

The text between the <body> tags is the text that will be displayed in your browser.

The text between the <b> and </b> tags will be displayed in a bold font.

### 1.2.1 HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension. We have used .htm in our examples. It might be a bad habit inherited from the past when some of the commonly used software only allowed three letter extensions.

With newer software we think it will be perfectly safe to use .html.

### 1.2.2 Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

### 1.2.3 Paragraphs

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

### 1.2.4 Line Breaks

The <br> tag is used when you want to end a line, but don't want to start a new paragraph. The <br> tag forces a line break wherever you place it.

```
<p>This <br> is a para<br>graph with line breaks</p>
```

The <br> tag is an empty tag. It has no closing tag.

### 1.2.5 Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

## 1.3 HTML Elements

This is an HTML element:

```
<b>This text is bold</b>
```

The HTML element starts with a start tag: <b>

The content of the HTML element is: This text is bold

The HTML element ends with an end tag: </b>

The purpose of the <b> tag is to define an HTML element that should be displayed as bold.

This is also an HTML element:

```
<body>
```

```
This is my first homepage. <b>This text is bold</b>
```



`</body>`

This HTML element starts with the start tag `<body>`, and ends with the end tag `</body>`.

The purpose of the `<body>` tag is to define the HTML element that contains the body of the HTML document.

### 1.3.1 Why do We Use Lowercase Tags?

We have just said that HTML tags are not case sensitive: `<B>` means the same as `<b>`. When you surf the Web, you will notice that most tutorials use uppercase HTML tags in their examples. We always use lowercase tags. Why?

If you want to prepare yourself for the next generations of HTML, you should start using lowercase tags. The World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags.

## 1.4 Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page.

This tag defines the body element of your HTML page: `<body>`. With an added `bgcolor` attribute, you can tell the browser that the background color of your page should be red, like this: `<body bgcolor="red">`.

This tag defines an HTML table: `<table>`. With an added `border` attribute, you can tell the browser that the table should have no borders: `<table border="0">`

Attributes always come in name/value pairs like this: `name="value"`.

Attributes are always added to the start tag of an HTML element.

### 1.4.1 Quote Styles, "red" or 'red' ?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

`name='John "ShotGun" Nelson'`

## CHAPTER TWO : JAVASCRIPT

### 2.1 Introduction

JavaScript has changed from being a language which improves web sites to a language which destroys them. This is because there are huge JavaScript sites which have thousands of scripts for download. These usually involve things which do not benefit a website at all, like status bar effects and scrolling text which do not add much to a website.

JavaScript must not be confused with Java. Java is a completely different programming language. It is usually used for text effects and games, although there are some JavaScript games around.

So why should you use JavaScript? Well, JavaScript can allow you to create new things on your website that are both dynamic and interactive, allowing you to do things like find out some information about a user (like monitor resolution and browser), check that forms have been filled in correctly, rotate images, make random text, do calculations and many other things.

In this tutorial I am assuming that you understand HTML.

### 2.2 Declaring JavaScript

Adding JavaScript to a web page is actually surprisingly easy! To add a JavaScript all you need to add is the following (either between the `<head></head>` tags or between the `<body></body>` tags - I will explain more about this later):

```
<script language="JavaScript">
```

```
JavaScript
```

```
</script>
```

As you can see the JavaScript is just contained in a normal HTML tag set. The next thing you must do is make sure that the older browsers ignore it. If you don't do this the code for the JavaScript will be shown which will look awful.

There are two things you must do to hide the code from older browsers and show something instead:

```
<script language="JavaScript">
```

```
<!--Begin Hide
```

```
JavaScript
```

```
// End Hide-->
```

```
</script>
```

`<noscript>`

## HTML Code

`</noscript>`

As you can see this makes the code look a lot more complex, but it is really quite simple. If you look closely you can see that all that has been done is that the JavaScript is now contained in an HTML comment tag. This is so that any old browsers which do not understand `<script>` will just think it is an HTML comment and ignore it.

Because of this the `<noscript>` tag was created. This will be ignored by browsers which understand `<script>` but will be read by the older browsers. All you need to do is put between `<noscript>` and `</noscript>` what you want to appear if the browser does not support JavaScript. This could be an alternative feature or just a message saying it is not available. You do not have to include the tag if you don't want anything shown instead.

### 2.2.1 Alerts

The first JavaScript command I will show you is:

`alert()`

This command will make a popup box appear. This can be useful for warning users about things or (in some cases) giving them instructions. It is used in the following way:

`alert('Text for alert box');`

In the example above I have used single quotations ' but you could use double quotations if you want to ". They work exactly the same way. The reason I use single quotes is because, later on, when you are using HTML code and JavaScript together you will need to use them and it is good to be consistent.

Here is the full JavaScript for the earlier example:

```
<script>
<!-- Start Hide

// Display the alert box
alert('This text is in an alert box');

// End Hide-->
</script>
```

This is placed between the `<head>` and `</head>` tags of the page. As you can see, I have used a comment tag as well as the alert box code. This makes your code more readable



but is not essential.

### 2.2.2 Variables

Variables in JavaScript, as in other computer languages, can be used to store information. For example I could tell the computer that the variable called:

`my_number`

should have the value:

3456

Variables can also contain text for example the variable:

`my_name`

could have the value:

David Gowans

Variables can be very useful for text or numbers that you repeat several times in a program, for doing calculations or for getting input from a user. Variables are declared as follows:

Number:

```
var my_number = 3456;
```

Strings (text):

```
var my_name = 'David Gowans';
```

As you can see a string is included in quotes (either single or double) but a number does not. If you include a number in quotes it will not be treated as a number. You may also notice that each line ends with a semicolon. This is standard JavaScript code to show the end of a line. Always remember to put it in.

When naming your strings you can use any word or combination of words as long as it is not already in use by JavaScript (so don't use `alert` as a variable name etc.). Do not include spaces, replace them with `_`.

### 2.2.3 Getting Information From The User

Once you have started using variables you will realize that it will be quite useful to get some information from the user. You can do this by using the:

#### 2.2.4 `prompt()`

First of all, the new prompt command is used. I set the variable `your_name` using it:

```
var your_name = prompt('Please enter your name', 'Your Name');
```

The text between the first set of quotes is what is written on the prompt box. The text between the second set of quotes is what is the default text for the input section of the box.

```
var output_text = welcome_text + your_name + closing_text;
```

As you can see this is much the same as adding 3 numbers together but, as these are strings they will be put one after the other (you could have also used quotes in here to add text and strings together). This added the text I had set as the `welcome_text` to the input I had received and then put the `closing_text` on the end.

Finally I displayed the `output_text` variable in an alert box with the following code:

```
alert(output_text);
```

which, instead of having text defined as the content for the alert box, places the string in the box.

### 2.2.5 `document.writeln`

This command is very useful as it will output information to a web page. I will start with a basic example of how to use it:

```
document.writeln('Hello there!');
```

This basically tells the JavaScript to write to the document (web page) on a new line the text `Hello there!`. The really useful bit of this is that you can tell the JavaScript to output text, HTML and variables. First of all I will show you how to output HTML:

```
document.writeln('<font face="Arial" size="5" color="red">Hello there!</font>');
```

This will display the following on the page:

Hello there!

As you can see, this allows you to just put standard HTML code into a web page using JavaScript. If you can't see a good reason for this it is not surprising at the moment but next I will introduce variables to the script.

## 2.3 Remote JavaScript

Now you have learnt how to use the `document.writeln` command you can now start using one of the most useful applications of JavaScript, remote scripting. This allows you to write a JavaScript in a file which can then be 'called' from any other page on the



web.

This can be used for things on your own site which you may like to update site-wide (like a footer on the bottom of every page) or something used on remote sites (for example newsfeed or some counters).

To insert a remote JavaScript you do the following:

```
<script language="JavaScript" src="http://www.yourdomain.com/javascript.js">
</script>
```

Which will then run the JavaScript stored at `http://www.yourdomain.com/javascript.js`. The .js file is also very simple to make, all you have to do is write your JavaScript (omitting the `<script>`, `</script>`, `<!--Start Hide and // End Hide-->` tags into a simple text file and save it as `something.js`.

If you want to include information for browsers which do not support JavaScript you will need to put the `<noscript></noscript>` tags in the HTML, not the JavaScript file.

There is one problem with using remote JavaScript which is that only the recent browsers support it. Some browsers which support JavaScript do not support Remote JavaScript. This makes it advisable not to use this for navigation bars and important parts of your site.

## 2.4 Link Events

A link event is a different way of including JavaScript on your page. Instead of using `<script>` tags in your HTML you can set JavaScript that will only be executed when certain things happen.

There are three ways of executing some JavaScript code in a link. They are:

- onClick
- onMouseOver
- onMouseOut

They can have many different uses but the most common is for image swaps (mouseover images).

### 2.4.1 onClick

onClick works in exactly the same way as a standard HTML link. It is executed when someone clicks on a link in a page. It is inserted as follows:

```
<a href="#" onClick="JavaScript Code">Click Here</a>
```

As you can see, one main difference is that the href of the link points to a #. This has nothing to do with the JavaScript, it just means that, instead of opening a new page, the link will not do anything. You could, of course, include a link in here and you would be able to open a new page AND execute some code at the same time. This can be used



you want to change the content of more than one browser window or frame at the same time.

### 2.4.2 onmouseover and onmouseout

onmouseover and onmouseout work in exactly the same way as onclick except for one major difference with each.

onmouseover, as the name suggests, will execute the code when the mouse passes over the link. The onmouseout will execute a piece of code when the mouse moves away from the link. They are used in exactly the same way as onclick.

## 2.5 Rollover Images

This is one of the main ways of using link events. If you have not seen rollover images before, they are images (usually used on navigation bars like the one at the top of this page) and when the mouse moves over the link it changes the image which is displayed.

This is done using a combination of the onmouseover and onmouseout events. To explain - when the mouse passes over the link you want the image to change to the new image, when it moves away from the link, the old picture should be displayed again.

The first thing you must do is edit the <img> tag you use to insert the image you want to change. Instead of just having something like this:

```

```

you would have the following:

```

```

The name for the image could be anything and, like the window names from the last part, is used to refer to the image later on.

Now you have given a name to the image you are using you will want to create the rollover. The first thing you must do is create the image for the rollover and save it. Then create a link round the image. If you don't want to have a link on the image you can still do a rollover by specifying the href as # which will make the link do nothing eg:

```
<a href="#"></a>
```

The following code will make a mouseover on your image (assuming you inserted the image as shown above and called your mouseover image homeon.gif):

```
<a href="index.htm" onmouseover="home_button.src='homeon.gif';"  
onmouseout="home_button.src='home.gif';"></a>
```

Firstly you are creating a standard link to index.htm. Then you are telling the browser that when the mouse moves over the link the image with the name home\_button will change to homeon.gif. Then you are telling it that when the mouse moves away from the link to change the image called home\_button to home.gif. Finally you are inserting an image called home\_button with an alt tag of 'Home' and no border round it.

onClick, onMouseOver and onMouseOut can be used with text links as well as images in exactly the same way as you did above. This, of course, means that you can create interesting effects by, when the mouse moves over an image, another image changes. This could be very useful for placing a description of a link on a page.

## 2.6 Status Bar

The status bar is the grey bar along the bottom of a web browser where information like, how much the page has loaded and the URL which a link is pointing to appears. You can make your own text appear in the status bar using the JavaScript you already know using the following code:

```
window.status='Your Text In Here';
```

You can include this in standard code, onClick, onMouseOver or onMouseOut. This allows you to do things like display a description of a link when the mouse moves over it.



## CHAPTER THREE : APACHE WEB SERVER

### 3.1 The WWW

The Internet has been around for a long time. More than 30 years now. But for most of that time, it was entirely the domain of geeks and hobbyists. The main reason for this was that it was hard to use.

In 1991, Tim Berners-Lee developed something that he called the World Wide Web, while working at CERN. His purpose was to give quick and easy access to documents for geographically distributed people collaborating on projects. Along with a lot of help from the standards community (and, notably, Roy Fielding), they defined HTTP, HTML, URLs, and the other necessary components of making the Web a reality. He then went off, and with the help of colleagues around the world, communicating via email, developed the CERN web server, and a simple Web client, which he dubbed a "browser." The name came about because there was very little of real value on the Web at that time, and all you ever really did was browse. Ironical that the name stuck!

#### 3.1.1 NCSA

As more and more people got involved in the project, it was several Universities that contributed to the project the most. From very early on, one of the front-runners was the National Center for Supercomputing Activities (NCSA) at the University of Illinois at Urbana Champaign (UIUC). NCSA started working on the NCSA HTTPd (HyperText Transfer Protocol Daemon).

Rob McCool wrote the original code for the NCSA HTTPd, and this code was distributed without charge to the community, for them to use, with the understanding that if they fixed bugs, or added features, that they would then contribute them back to Rob to put into future versions.

### 3.2 Introduction - What is Apache

The Apache web server is the best, and most preferred, HTTP server software in use on the Internet today, and it was written entirely as a volunteer project, by volunteer programmers, in their spare time. The Apache web server project is more than just a piece of software. That, in itself, is astonishing. That is, it is to people that are not familiar with the Open Source methodology, and Open Source projects like Linux, Perl, Sendmail, and a variety of others. The interesting thing about these volunteer-written, free software packages is that most of us, and our businesses, rely heavily on them, whether we are aware of it or not.

Before diving directly into talking about what Apache is, it is useful to talk about where Apache came from, and how it came to be.

#### 3.2.1 The Apache Server

When Rob left the project, it left a problem. There were still a lot of people using his code, and actively making patches to the code, but there was no longer anyone collecting those patches.

In 1995, Brian Behlendorf and a small group of other developers started collecting these patches in a central repository. Brian got some space donated on a server, and set up a CVS tree so that developers could check in patches. And in April of 1995, they released the first official release (Version 0.6.2), which was given the name Apache, because it was "a patchy server".

The Apache Group, as they were known at that time, had no formal organizational structure, never met, communicated only over email, and worked entirely in their free time, on a volunteer basis. Early the next year, Apache passed NCSA as the most widely used server on the Internet, and is now used on more than 60% of all web servers on the Internet.

### **3.2.2 Apache's Architecture**

Since the 1.0 release of Apache (December 1, 1995) Apache has has a modular design. The core of the server is very light-weight, and all other functions are implemented as modules that plug in to the core. This means that you can keep the size of the executable down by leaving out functionality that you don't need. It also means that if there is some functionality missing that you do need, you can write your own custom module to plug into the core.

### **3.2.3 More Recent History**

In the last few years, Open Source has been getting a lot of press, because of Linux, Perl, and Apache. In 1998, IBM decided to abandon development of a web server engine to go into WebSphere - an application server for the web - and use Apache instead. This decision, along with Netscape's decision to release the source code for the Netscape browser, earlier that same year, showed the business world that Open Source was more than just a lot of long-haired anti-establishment types out to bring down the software industry, but that it was actually a good business model. It produces code more quickly, and that code is more reliable, because, in the words of Eric Raymond, with enough eyes, all bugs are shallow.

In June of 1999, The Apache Software Foundation was officially incorporated in the state of Delaware. The ASF has a much broader mission than just the Apache HTTP server, and has several other projects that exist under the larger umbrella of the ASF. The stated goals of the ASF are:

- provide a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure;
- create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit;
- provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation's projects; and,
- protect the 'Apache' brand, as applied to its software products, from being abused by other organizations.

Some of the better-known projects under the ASF are the Apache web server, `mod_perl`, `mod_php`, and Jakarta.

### **3.2.4 The Future of Apache**



At ApacheCon in Orlando, back in March, Apache 2.0 was released. This is largely a rewrite from earlier versions, and uses a threading model that will increase performance substantially on most platforms. As of this writing, version Alpha 6 of the 2.0 server has been released.

The Apache Group, as mentioned above, has become the Apache Software Foundation, and continues to take on new projects that seem to fit the larger vision that the ASF has for the future. Open Source, and open standards, produce better software. In the end, this makes life better for all of us, and we should support the ASF in all its endeavors, if only for purely selfish reasons.

### 3.2.5 Support for Apache

As an Open Source software product, Apache falls prey to the myth of no support.

There are two main ways to obtain support for Apache. First, there's the traditional email and Usenet methods. There are a variety of mailing lists on which you can obtain support for Apache. And there are two main Usenet groups -

comp.infosystems.www.servers.unix and, for those running Apache on Windows, comp.infosystems.www.servers.mswindows

You can find information about the ``official" Apache mailing lists on the Apache.org web site.

Secondly, there's also commercial support for Apache, available through Covalent Technologies. Covalent offers support contracts for Apache, and they also have add-on products for Apache, such as Raven SSL. And the author of Comanche (which we'll discuss later) works at Covalent.

## 3.3 Obtaining and Installing Apache

Apache is available as source code, and is probably available as a binary installation for your operating system, unless you are running something truly arcane and rare. And, of course, if you are, you can still get the source code, and compile it yourself.

### 3.3.1 Hardware/Software Requirements

Apache runs on anything. Almost. It will almost certainly run on whatever you have. I've run Apache on a 386 with 4 MB of RAM. And I've run it on a 4-processor machine with 1 GB of RAM. It was happy both places. The Apache.org web site does not list any hardware requirements. It will run on any hardware that runs the supported operating systems.

Apache will run on any flavor of \*nix, and also on Microsoft Windows (95, 98, NT, 2000), Mac, and OS/2.

Compiling and installing

Most of the settings for your server, governing how it will operate, are done at the configuration stage, when you modify the configuration files that the server loads when it starts up. However, due to the modular architecture of Apache, a lot also depends on what modules you enable when you compile the server. The available configuration directives depend on the modules that are loaded.

You can either compile your server the quick, easy way, and get a default installation with the most common functionality, or you can get in there and pick and choose what you actually want.

The simple way

The installation process for Apache is really simple for most folks. If you are just wanting to set up a simple web site to do the normal things like serve web pages, and maybe do some CGI, the installation process looks like this:

```
tar -zxf apache_1.3.12.tar.gz
cd apache_1.3.12
./configure --prefix=/usr/local/apache
make
make install
/usr/local/apache/bin/apachectl start
```

Assuming you have a reasonably fast machine, this entire process does not take much more than 10 or 15 minutes, and you have a functioning web site. The configure process figures out reasonable settings for your system, and so the configuration files will have reasonable things in them so that you can immediately start serving web pages from your server. The --prefix setting tells the configure process where you want to install the server. /usr/local/apache is the normal place to do this, but if you want to put it somewhere else, just specify that on the command line:

```
./configure --prefix=/home/rbowen/devserver
```

apachectl is a handy tool that Apache installs to make it simple to start, stop, and restart the server, as well as some other handy functionality. More about this a little later.

### 3.3.2 Advanced Installation

If you're like me, you are not satisfied with the default installation. It does not have all the modules that I want, it has some stuff that I'd just as soon leave out, and there are some things I just do differently because I do them differently. I'm strange that way.

There are two ways to handle this that I'm going to talk about. First, you can actually edit the configuration file, and specifically choose what you want to compile into the server. Or, you can just throw everything in, but do it in such a way that you can go back and add and remove stuff at your leisure. I tend to go with the latter approach, but the former approach gets more coverage in the docs, and so is used more frequently. You are advised to use the simple method above the first few times you install Apache. Also, in version 2.0, there will only be one installation method, and it will look more like the quick easy method above, than like these methods here.

### 3.3.3 Editing The Configuration Scripts

In our previous example, we ran a script called configure (``small-c configure'') in the main Apache directory. In this method, we're going to go down into the src/ directory and actually look at the configuration files. After all, the motto of Linux is ``do it yourself.''

The process starts out the same:

```
tar -zxf apache_1.3.12.tar.gz
```

But rather than just going into the apache\_1.3.12 directory, you need to go down into the src directory:

```
cd apache_1.3.12/src
```



Then, using your favorite editor, edit the file Configuration ("big-C Configuration") and comment, or uncomment, the lines that refer to options that you are interested in. If you don't see a file called Configuration, copy the file Configuration.tmpl to Configuration, and use that as your template. Once you have gone through and made sure that you have everything that you want in there, save the file, and run the following:

```
./Configure  
make  
make install
```

The simple method, which we talked about first, is doing all of this for you behind the scenes. If you run "small-c" configuration, you will have a file called Configuration.apaci created for you, which will then get used in this configuration step.

### 3.3.4 Dynamic Shared Objects (DSO's)

I get tired of rebuilding and reinstalling my web server every time I want to add in a new module, or when I decide to take one out, because I never really use it. This is where shared objects are handy. A shared object is something that gets loaded dynamically by a process when it needs it. This saves you from having to compile that code into the program executable, which, in turn, makes the executable smaller, and load up faster. By making your Apache modules into shared objects, you can build everything into your server, but only actually use the parts that you want at any one time, and leave out everything else that you're not using.

On Windows, these are things called "dynamic link libraries", or DLLs. On Linux, they are called shared objects, or .so files.

In order to enable shared objects, you have to compile a module called mod\_so, which, in turn, loads all the modules that you have compiled as shared objects. mod\_so itself cannot be shared object, of course, because there would be no way to load it. Chicken, egg.

So, to build your Apache server to use shared objects, run the following commands:

```
./configure --prefix=/path/to/apache \  
--enable-module=most \  
--enable-shared=max
```

(You can either literally type those \ characters, or just put this command all on one line and omit the \s)

What does this command do? Well, it compiles all of the modules that ship with Apache, except those that are considered experimental or unstable, and enables them all as DSO's.

This means that Apache will be loading up a bunch of modules that you don't really want, so you need to edit the configuration file and comment out those modules that you're not really going to use.

But it also means that if you want to add in a particular module, you can do so by putting it in the configuration file, rather than having to recompile your server from source. This is particularly handy if you change your server configuration a lot, or when you are testing out different configurations to see what it is that you want.

A server that is loading all the modules dynamically, rather than having those modules compiled in, takes a little longer to start up, but this penalty is paid only at server start, and after that the servers run at the same speed. That is, a server running modules as DSO's does not run any slower.

### 3.4 Configuring

Once you've compiled and installed your server, you need to configure it for your particular environment. Many of the configuration directives got set when you ran `configure` (or `Configure`), and so the server should work correctly immediately. However, you will probably want to change some things, since the default installation is very generic, and not precisely suited to your needs.

Apache, unlike most of its competitors in the web server market, lets you configure everything, down to the smallest detail. And if there's really something that you want to configure that you can't, you have the source code, so you can change it if you are so inclined.

#### 3.4.1 Configuration Files

The configuration for your Apache server is located in a file called `httpd.conf`, which is usually located at `/usr/local/apache/conf/httpd.conf`.

Note that if you installed Apache with a RPM (don't do that!) then the files will be in bizarre places that have no relation to logic. Uninstall the RPM, and install from source. It's a simple process, and reduces your pain in the long run.

Note: I made a comment like the above in one of my articles on [ApacheToday.com](http://ApacheToday.com), and got thoroughly chastized for it by some Red Hat fanatics that read the article. While I found their comments, and their reasons for their comments, to be rather amusing, I should emphasize that this is just my opinion, and should not be taken as some sort of transcendent truth. If you really want to spread your files all over your file system, go right ahead. You might notice, however, that a default installation of Apache puts everything in `/usr/local/apache`, so it's a safe bet that the Apache developers agree with me on this one.

The format of `httpd.conf` is very simple.

There are comments, which consist of a hash sign (#) at the beginning of a line:

# Based upon the NCSA server configuration files originally by Rob McCool.

There are directives, which look like a name, followed by a value:

ServerAdmin webmaster@rcbowen.com

There are sections, or containers, which look rather like HTML tags:

<Directory /usr/local/apache/cgi-bin>

AllowOverride None

</Directory>

Sections can contain directives, and those directives apply to the the resources defined by the container definition. In the above example, the `AllowOverride` directive will apply to files located in the specified directory.

You can edit these configuration files with your favorite text editor. You need to restart the server when you are done editing the configuration files in order for the new configuration to take effect.

You can use the `apachectl` script to test your configuration file to make sure that you did not make any errors.

`/usr/local/apache/bin/apachectl configtest`

More about this below.



### 3.4.2 Comanche

One of the battles that \*nix continually has to fight is the notion that it's hard to use. Much of this notion comes from the fact that everything you want to use on \*nix has a configuration file, and every configuration file has a different format. Learning all these different formats is a pain, and it's so easy to get it wrong. Sendmail is one of the worst offenders in this arena, but even something as simple as Apache gets difficult to configure. Its modular architecture means that it can be extended forever, and every extension has its own configuration directives. This can be a little overwhelming. Daniel Lopez took on this problem as his Master's thesis, and developed Comanche - the Configuration Manager for Apache. Comanche is a graphical configuration tool, written in Tcl, which lets you configure Apache in a more intuitive interface. It tells you what each directive means, and asks you questions that make sense. Your answers are put back into the configuration files in a format that Apache can understand. Comanche can also be used to configure other applications, such as Samba, which have text configuration files. There is not yet a plug-in for configuring Sendmail, but this is something that Daniel is frequently asked for, so perhaps there will be some day. And you can write your own extensions to Comanche to configure anything that has a text configuration file.

### 3.4.3 Starting, Stopping, Restarting

There are a variety of ways to control your Apache server. We'll focus on a script that ships with Apache, called `apachectl`, which does a few other things in addition to just starting, stopping, and restarting.

`apachectl`

`apachectl`, which presumably stands for "Apache control", is located in the `bin` directory of your Apache installation. It is a shell script which does many of the things that you'll want to do in controlling your Apache server. It can be run with any of the following arguments:

`start`

Starts the server.

`stop`

Stops the server.

`restart`

Restarts the server, if running, by sending a `SIGHUP`. If the server is not running, starts it.

`fullstatus`

Displays the full status of the server. Requires that `mod_status` is enabled, and that `lynx` is installed.

`status`

Displays a brief status report for the server. Requires that `mod_status` is enabled, and that `lynx` is installed.

`graceful`

Does a graceful restart by sending a `SIGUSR1`, if the server is running. If the server is not running, it will start it. A graceful restart has the advantage over a simple restart in that child processes that are currently serving content will be permitted to complete their current connection before they are killed.

`configtest`

Reads the configuration file and parses it for syntax errors.

help

Displays usage information about the apachectl script.

Starting your Apache server on system restart

Linux has a process for starting processes on system startup. This consists of a directory /etc/rc.d containing scripts for each of the processes that you want to start.

If you place a file in /etc/rc.d, called rc.httpd, it will be run on server startup. rc.httpd should contain the following command:

```
/usr/local/apache/bin/apachectl start
```

If you're running Red Hat, or Mandrake, or one of the other Linuxes that look like them, you'll find that there are a number of subdirectories of /etc/rc.d that look like rc2.d, rc3.d, and so on, which contain the startup scripts for all your various services. Actually, symlinks to them. On these systems you should create a file at /etc/rc.d/init.d/httpd, containing the command above. You should then create links to it from the directories rc3.d and rc5.d. Each of those directories corresponds to a runlevel. You'll usually be in either runlevel 3 or 5, so that's when you want to start Apache.

### 3.5 Integrating Apache With The Rest of Your business

The common wisdom is that every company needs a web site, because every company needs a web site. And so lots of companies have web sites which are nothing more than a electronic sales brochure.

With more and more people online every day, many users will expect to get just as good service from your web site as they would in person, or over the phone. In fact, the expectation is often higher. After all, this is a computer. They should be able to get direct access to the answers that they need, and it should be instantaneous.

#### 3.5.1 CGI

The most common way to tie your web site to you database or other processing is with CGI programs. The Common Gateway Interface is a protocol to let your web server serve dynamically generated content from some process running on your server. Of course, I've oversimplified it in that statement, but that's probably sufficient for our purposes.

A CGI program is a program written in any language you like, which formats its output in a certain way so that your browser can understand it. This allows you to write programs to put any of your existing databases onto your web site, and interact with your online customers in realtime, directly on your web site. You can let the customer customize their experience of your web site.

There are a plethora of good CGI tutorials on the web. (and even more bad ones!) But the basic concepts are pretty simple

#### 3.5.2 MIME Headers

Any output that your CGI program produces must be preceeded by a MIME header that tells the client (the browser) what sort of output they are receiving. This will look something like:

```
content-type: text/html
```



HTTP headers are followed by a blank line, which is how the client knows that the headers are done, and the next things that it sees are the body of the document. If you were to write a CGI program in Perl, for example, this would look like:

```
print "content-type: text/html\r\n\r\n";
```

`\r\n` is called a `crlf`, which is short for "carriage return line feed." Frequently, you will see CGI programs that have just `\n`, rather than both, but it is more correct to use both. This used to be more of a big deal than it is now. Most web browsers are quite happy to accept one or the other.

### 3.5.3 Reading Client Input

Input from the client comes in to your program on `STDIN`. This means that you can read client input as though it was coming from the command line or from the keyboard.

Of course, most languages that you are likely to use for CGI programming have libraries readily available that will handle most of the mundane details of CGI programming for you, and leave you to do your work.

For example, in Perl, there is the `CGI.pm` module, and a few others, such as the `CGI_Lite.pm` module, that handle such things as reading form input and managing cookies, and in the case of `CGI.pm` generating your headers and output

In C, there are libraries available from Tom Boutell at [Boutell.com](http://Boutell.com) which provide similar functionality from C.

Perl is the language of choice for CGI programming, because it is very conducive to the sort of rapid prototyping and development that is often demanded by the web.

Output in HTML (usually)

Since your output is going to a browser, you will almost always want to have your output in HTML. Occasionally, you'll want your output to be a gif image, or plain text.

Example CGI program

The following is an example CGI program written in Perl. It does not actually do anything useful, but it gives you an idea of what is the minimum necessary requirement for a CGI program.

```
#!/usr/bin/perl
print "content-type: text/html\r\n\r\n";
print "<b>Hello, World!</b>";
```

Not very exciting, is it?

Rather than provide a lot of example CGI programs, I'd encourage you to look at all the resources at the end of this paper for examples.

### 3.5.4 `mod_perl`

Something that you will eventually discover when using CGI is that it is slow. This has nothing to do with the quality of the code that you write, but is intrinsic to CGI. The problem is that every time a client requests a resource that involves running a CGI program, Apache has to launch that program. That takes a lot of time. This is the case whether the program is a Perl script or a compiled binary executable. Almost all of your time will be spent in the startup of that program, not in the actual run time of the program. There are programs with intensive database access, where this will not be the case, but they are in the minority.

There are a number of different technologies that address this problem. Most of them involve having your CGI programs somehow cached, so that when they are invoked,

you don't have to pay that startup time, because they are already there in memory, ready to go.

Perhaps the most popular of these solutions is `mod_perl`. `mod_perl` is an Apache module that significantly enhances the performance of Perl CGI programs. It has other benefits, such as the ability to write Apache modules in Perl, but it is primarily used as a CGI enhancer.

Your Perl CGI programs are compiled, and kept in memory, so that every time the resource is requested, there is no time spent launching the Perl interpreter, or loading your program from disk.

`mod_perl` is extremely memory-intensive, since all this code is stored in memory. But the performance enhancements are several orders of magnitude, producing a very noticeable speed increase.

### **3.5.5 SSL and E-Commerce**

Even better than telling your customers about your business, is actually doing business online. You can put your catalog online, and let customers order, and pay for, your merchandise directly on your web site.

The one problem is that people are rather picky about who they give their credit card information to. And they are extremely reluctant to type it in and submit it across the Internet without some assurance that what they are doing is secure.

By default, data that is passed to web servers is "in the clear", meaning that it is not encrypted, and anyone that is watching the wire would be able to see anything that went past. Like, for example, your credit card number. Even when you are in a password protected area on a web site, the username and password, and any data exchanged, is all passed in the clear.

One way around this is SSL. SSL, which stands for Secure Socket Layers, is a technology that encrypts traffic between the server and the client using a private key/public key technique. That means that only the people on the two ends can understand it. And even if someone were to intercept the entire message, they would not be able to decrypt it.

There are a number of SSL implementations that run on top of Apache. Two of the better known ones are Raven, from Covalent (<http://www.covalent.com/raven/ssl/>), and Stronghold (<http://www.c2.net/products/sh2/>). These are both commercial products.

The Open Source alternative is `mod_ssl` (<http://www.modssl.org/>), which runs in conjunction with OpenSSL (<http://www.openssl.org>).

With the recent changes in the crypto laws, there's a good chance that `mod_ssl` will ship as one of the standard Apache modules in future releases.

For more information on SSL, you should attend one of the SSL talks here at ApacheCon.

### **3.5.6 Authentication**

Authentication is the process of verifying that you are who you say you are. This is usually accomplished by requesting some variety of username and password. There are a number of different implementations of this for use with Apache.

### **3.5.7 `mod_auth`**



The "standard" Authorization technique is to use HTTP authentication provided by the Apache module called `mod_auth`. `mod_auth` is part of a standard installation of Apache, and is turned on by default.

To enable authentication for a particular directory, you need to do several things.

Create a password file

Using the `htpasswd` utility that comes with Apache, you need to create a password file, which tells apache what password is required for what username, in order to get to the resources in question.

The help for `htpasswd` says the following:

Usage:

```
htpasswd [-cmdps] passwordfile username
htpasswd -b[cmdps] passwordfile username password
-c Create a new file.
-m Force MD5 encryption of the password.
-d Force CRYPT encryption of the password (default).
-p Do not encrypt the password (plaintext).
-s Force SHA encryption of the password.
-b Use the password from the command line rather than prompting for it.
```

On Windows and TPF systems the '-m' flag is used by default.

On all other systems, the '-p' flag will probably not work.

The `htpasswd` utility is (usually) located in `/usr/local/apache/bin`.

So, for example, to create a new password file, you would type:

```
htpasswd -c htpasswd rbowen
```

You will then be asked for the password that you want that user to have, and then you'll be asked to type it again to confirm it.

To add a password to an existing file, type the same command, but without the `-c`.

Create a group file

If you want to allow more than one user to have access to a particular resource, you can create a group of users. This is done by creating a group file which lists group names and the members in those groups. A line in the group file might look like this:

```
TCG: rbowen sungo chad tom
```

Put your files somewhere safe

You should store these files (the password file and the group file) somewhere outside of the document directory, so that they cannot be downloaded for leisurely off-line hacking.

Create a `.htaccess` file pointing at these files

In the directory that you want to protect, create a file called `.htaccess`, containing something like the following:

```
AuthName "Members Only"
AuthType Basic
AuthUserFile /path/to/htpasswd
AuthGroupFile /path/to/htgroup
Require group TCG
```

The `AuthName` is the string that appears in the authentication dialog that pops up when you visit a protected area.

`AuthType` is the method of authentication. It is one of Basic or Digest, but Basic is the only one of these methods that is widely implemented in browsers, so you should probably stick to that.

`AuthUserFile` and `AuthGroupFile` refer to the locations of the user file and group files that we created in the steps above.

Require is the directive that tells Apache what user(s) or group(s) can get the content specified. You can Require a particular user, or several users, rather than a particular group:

Require user Tom,Dick,Larry

The configuration detailed above will protect all files in a particular directory and all subdirectories thereof. You can also protect individual files with a <Files> section. See the documentation for more details.

mod\_auth\_db, mod\_auth\_mysql, etc

There are a variety of other modules that allow you to authenticate against usernames and passwords stored in a variety of other places, from DBM files, to MySQL databases, to Oracle databases, to Netware directory services. And a variety of other things. There are modules for authentication against a NT domain, or against Lotus Notes. Check out [modules.apache.org](http://modules.apache.org) for an impressive listing of Apache modules for these and other uses.

### 3.5.8 Log Files

Apache writes two log files as it runs - the access\_log, which keeps a record of every request that your server receives, and the error\_log, which keeps track of everything that goes wrong, or other less urgent information, such as server startup, stop, and restarts.

access\_log

access\_log, by default, is stored in the common log format, which contains the following information:

Address of the client

The address of the remote machine requesting content from your server. This is usually just the IP address, but if you turn HostNameLookups on, this will be, whenever possible, the fully qualified domain name of the client.

ident

The information returned by ident, or other similar lookup. This used to frequently actually contain the email address of the remote user, but this practice stopped as soon as it was realized that people were collecting this information for spam lists.

Username

If the resource requested was password protected, this field will contain the username that was used to gain access.

date/time

The date and time of the request.

Request

The first line of the request that was made to the server

Status

The return code the server returned to the client. 2xx messages mean everything went well. 3xx messages mean that the server redirected the request. 4xx messages mean the user did something wrong. 5xx messages mean that the server did something wrong.

Bytes sent

How many bytes were actually sent to the client.

error\_log



The `error_log` contains errors and various other messages that the server generates during operation. This is particularly useful for troubleshooting CGI programs that are not behaving as expected.

#### Custom log files

With the `LogFormat` and `CustomLog` directives, you can create your own log files that contain whatever information you'd like to collect. See the Apache documentation for more details on generating these log formats.

#### Contributing to the project

Apache is a volunteer-driven project. That means that it relies largely on the users to contribute patches, suggestions, bug reports, and comments.

And big piles of money, of course.

If you think that you have any of the above, and you want to contribute them, here's how to go about it.

Within each project in the Apache Software Foundation, development is completely autonomous from the Foundation as a whole. Each project is left to manage affairs as best suits that project. Each project has its own web site off of the main [www.apache.org](http://www.apache.org) web site, and you can usually find information on those sites about contributing code or documentation patches.

### 3.6 A Conclusion for Apache

Apache is the web server that you need to be using. There's really no question about it. 60% of all web site administrators can't be wrong.

When your web site is becoming such an important, integral part of your business, you really can't afford to be running anything but the best.

## CHAPTER FOUR PERSONAL HOME PAGE (PHP)

### 4.1 What is PHP?

PHP is a server-side scripting language designed specifically for the web. Within an HTML page, we can embed PHP code that will be executed each time the page is visited. Our PHP code is interpreted at the Web server generates HTML or other output that the visitor will see.

PHP was conceived in 1994 and was originally the work of one man, Rasmus Lerdorf. It was adopted by other talented people and has gone through three major rewrites to bring us the broad, mature product we see today. As of January 2001, it was in use on nearly five million domains worldwide, and this number is growing rapidly.

PHP is an Open Source product. You have access to the source code. You can use it, alter it, and redistribute it all without charge.

PHP originally stood for Personal Home Page, but was changed in line with the GNU recursive naming convention (GNU = Gnu's Not Unix) and now stands for PHP Hypertext Processor.

### 4.2 Writing PHP

Writing PHP on your computer is actually very simple. You don't need any special software, except for a text editor (like Notepad in Windows). Run this and you are ready to write your first PHP script.

#### 4.2.1 Basic PHP Syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with `<?` and end with `?>`.

However, for maximum compatibility, we recommend that you use the standard form (`<?php`) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: `echo` and `print`. In the example above we have used the `echo` statement to output the text "Hello World".



### 4.2.2 Comments in PHP

In PHP, we use // to make a single-line comment or /\* and \*/ to make a large comment block.

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
block
*/
?>
</body>
</html>
```

### 4.2.3 Declaring PHP

PHP scripts are always enclosed in between two PHP tags. This tells your server to parse the information between them as PHP. The three different forms are as follows:

```
<?
PHP Code In Here
?>
```

```
<?php
PHP Code In Here
php?>
```

```
<script language="php">
PHP Code In Here
</script>
```

All of these work in exactly the same way but in this tutorial I will be using the first option (<? and ?>). There is no particular reason for this, though, and you can use either of the options. You must remember, though, to start and end your code with the same tag (you can't start with <? and end with </script> for example).

The first PHP script you will be writing is very basic. All it will do is print out all the information about PHP on your server. Type the following code into your text editor:

```
<?
phpinfo();
?>
```

As you can see this actually just one line of code. It is a standard PHP function called phpinfo which will tell the server to print out a standard table of information giving you

information on the setup of the server.

This is very important. As with many other scripting and programming languages nearly all lines are ended with a semicolon and if you miss it out you will get an error.

Almost all of this is instantly recognizable to you as plain old HTML code -- all except that one curious line in the middle: `<?php phpinfo(); ?>` That's the PHP code.

A PHP code snippet is surrounded by an opening and a closing delimiter. The opening delimiter is `<?php` and the closing is `?>` the code in between is PHP code. In this example we are calling the PHP function "phpinfo". The function name is followed by opening and closing parentheses which surround any parameters being passed to the function -- in this case there are none. The function call is terminated with a semicolon. Go ahead and create a file containing this code -- the file name should end with a .php suffix -- send it up to your server and pull it up in your web browser just like you would any other web page. you should see lots of interesting info about the server system and PHP itself. Take a look at the source of the resulting page (View Source). You'll notice that the HTML from your original page is still there, but the PHP code has been replaced by all the information you see. That's basically how the engine works. The PHP code you write is replace in the resulting web page by the results of running the PHP code. Nobody gets to see your actual PHP code -- only the result of it running. That's significant, as I imagine you already realize!

For a little further note about PHP syntax; the code snippet above, contained between the beginning and ending delimiters is similar to HTML tags in appearance. It is in fact called a PHP tag. There are four forms of PHP tags. The one we've used here is called an XML style tag and is the one we will continue to use throughout this tutorial. It is the most common form. another style is called script style and will also look somewhat familiar to you if you have written any other script code such as JavaScript. This example is exactly equivalent to our snippet above:

```
<script language=php>
phpinfo();
</script>
```

The other two forms are called the short style of tag and the ASP style. Both of these require special settings in the PHP configuration files. They are less commonly used and I will only provide this one example of each in this series. Here is the same code in the Short style and ASP style, respectively:

```
<? phpinfo(); ?>
<% phpinfo(); %>
```

As I previously mentioned, each statement in your PHP code is ended with a semicolon. Leaving this semicolon off is a common syntax error and when something isn't working the way you expect it to should be one of the first things you check. Whitespace (spaces, tabs and newlines) are ignored in the syntax of PHP. These three examples are completely equivalent:

```
<?php phpinfo(); ?>
<?php  phpinfo();  ?>
<?php
phpinfo();
?>
```

#### 4.3 Variables

As with other programming languages, PHP allows you to define variables. In



PHP there are several variable types, but the most common is called a String. It can hold text and numbers. All strings begin with a \$ sign. To assign some text to a string you would use the following code:

```
$welcome_text = "Hello and welcome to my website.";
```

This is quite a simple line to understand, everything inside the quotation marks will be assigned to the string. You must remember a few rules about strings though:

Strings are case sensitive so \$Welcome\_Text is not the same as \$welcome\_text  
String names can contain letters, numbers and underscores but cannot begin with a number or underscore

When assigning numbers to strings you do not need to include the quotes so:

```
$user_id = 987
```

would be allowed.

### 4.3.1 Outputting Variables

To display a variable on the screen uses exactly the same code as to display text but in a slightly different form. The following code would display your welcome text:

```
<?
$welcome_text = "Hello and welcome to my website.";
print($welcome_text);
?>
```

As you can see, the only major difference is that you do not need the quotation marks if you are printing a variable.

### 4.3.2 Formatting Text

Everything is just output in the browser's default font. It is very easy, though, to format your text using HTML. This is because, as PHP is a server side language, the code is executed before the page is sent to the browser. This means that only the resulting information from the script is sent, so in the example above the browser would just be sent the text:

Hello and welcome to my website.

This means, though, that you can include standard HTML markup in your scripts and strings. The only problem with this is that many HTML tags require the " sign. You may notice that this will clash with the quotation marks used to print your text. This means that you must tell the script which quotes should be used (the ones at the beginning and end of the output) and which ones should be ignored (the ones in the HTML code).

Change the text to the Arial font in red. The normal code for this would be:

```
<font face="Arial" color="#FF0000">
</font>
```

As you can see this code contains 4 quotation marks so would confuse the script. Because of this you must add a backslash before each quotation mark to make the PHP script ignore it. The code would change to:

```
<font face=\"Arial\" color=\"#FF0000\">
</font>
```

You can now include this in your print statement:

```
print("<font face=\"Arial\" color=\"#FF0000\">Hello and welcome to my
website.</font>");
```

which will make the browser display:

Hello and welcome to my website.

because it has only been sent the code:

```
<font face="Arial" color="#FF0000">Hello and welcome to my website.</font>
```

## 4.4 PHP Operators

This section lists the different operators used in PHP.

### 4.4.1 Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8	1 2



		10%2	0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

#### 4.4.2 Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

#### 4.4.3 Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

#### 4.4.4 Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
	or	x=6 y=3 (x==5    y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

#### 4.5 Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

if...else statement - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

elseif statement - is used with the if...else statement to execute a set of code if one of several condition are true

#### 4.5.1 If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if...else statement.

Syntax

```
if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
```

#### Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
else
echo "Have a nice day!";
?>
</body>
</html>
```

If more than one line should be executed when a condition is true, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
$x=10;
if ($x==10)
{
echo "Hello<br />";
echo "Good morning<br />";
}
?>
</body>
</html>
```

#### 4.5.2 The Else If Statement

If you want to execute some code if one of several conditions are true use the If...Else statement with Else If



### Syntax

```
if (condition)
code to be executed if condition is true;
else if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
```

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
elseif ($d=="Sun")
echo "Have a nice Sunday!";
else
echo "Have a nice day!";
?>
</body>
</html>
```

## 4.6 PHP String Processing

The string functions allow you to manipulate strings.

### 4.6.1 PHP String Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
<u>addslashes()</u>	Returns a string with backslashes in front of the specified characters	4
<u>addslashes()</u>	Returns a string with backslashes in front of predefined characters	3
<u>bin2hex()</u>	Converts a string of ASCII characters to hexadecimal values	3
<u>chop()</u>	Alias of rtrim()	3
<u>chr()</u>	Returns a character from a specified ASCII value	3
<u>chunk_split()</u>	Splits a string into a series of smaller parts	3
<u>convert_cyr_string()</u>	Converts a string from one Cyrillic character-set to another	3
<u>convert_uudecode()</u>	Decodes a uuencoded string	5
<u>convert_uuencode()</u>	Encodes a string using the uuencode algorithm	5
<u>count_chars()</u>	Returns how many times an ASCII character occurs within a string and returns the information	4



<u>crc32()</u>	Calculates a 32-bit CRC for a string	4
<u>crypt()</u>	One-way string encryption (hashing)	3
<u>echo()</u>	Outputs strings	3
<u>explode()</u>	Breaks a string into an array	3
<u>fprintf()</u>	Writes a formatted string to a specified output stream	5
<u>get_html_translation_table()</u>	Returns the translation table used by <u>htmlspecialchars()</u> and <u>htmlentities()</u>	4
<u>hebrew()</u>	Converts Hebrew text to visual text	3
<u>hebrevc()</u>	Converts Hebrew text to visual text and new lines (\n) into  	3
<u>html_entity_decode()</u>	Converts HTML entities to characters	4
<u>htmlentities()</u>	Converts characters to HTML entities	3
<u>htmlspecialchars_decode()</u>	Converts some predefined HTML entities to characters	5
<u>htmlspecialchars()</u>	Converts some predefined characters to HTML entities	3
<u>implode()</u>	Returns a string from the elements of an array	3
<u>join()</u>	Alias of <u>implode()</u>	3
<u>levenshtein()</u>	Returns the Levenshtein distance between two strings	3
<u>localeconv()</u>	Returns locale numeric and monetary formatting information	4
<u>ltrim()</u>	Strips whitespace from the left side of a string	3
<u>md5()</u>	Calculates the MD5 hash of a string	3
<u>md5_file()</u>	Calculates the MD5 hash of a file	4
<u>metaphone()</u>	Calculates the metaphone key of a string	4
<u>money_format()</u>	Returns a string formatted as a currency string	4
<u>nl_langinfo()</u>	Returns specific local information	4
<u>nl2br()</u>	Inserts HTML line breaks in front of each newline in a string	3
<u>number_format()</u>	Formats a number with grouped thousands	3
<u>ord()</u>	Returns the ASCII value of the first character of a string	3
<u>parse_str()</u>	Parses a query string into variables	3
<u>print()</u>	Outputs a string	3
<u>printf()</u>	Outputs a formatted string	3
<u>quoted_printable_decode()</u>	Decodes a quoted-printable string	3
<u>quotemeta()</u>	Quotes meta characters	3
<u>rtrim()</u>	Strips whitespace from the right side of a string	3
<u>setlocale()</u>	Sets locale information	3
<u>sha1()</u>	Calculates the SHA-1 hash of a string	4
<u>sha1_file()</u>	Calculates the SHA-1 hash of a file	4
<u>similar_text()</u>	Calculates the similarity between two strings	3
<u>soundex()</u>	Calculates the soundex key of a string	3
<u>sprintf()</u>	Writes a formatted string to a variable	3



<u>sscanf()</u>	Parses input from a string according to a format	4
<u>str_ireplace()</u>	Replaces some characters in a string (case-insensitive)	5
<u>str_pad()</u>	Pads a string to a new length	4
<u>str_repeat()</u>	Repeats a string a specified number of times	4
<u>str_replace()</u>	Replaces some characters in a string (case-sensitive)	3
<u>str_rot13()</u>	Performs the ROT13 encoding on a string	4
<u>str_shuffle()</u>	Randomly shuffles all characters in a string	4
<u>str_split()</u>	Splits a string into an array	5
<u>str_word_count()</u>	Count the number of words in a string	4
<u>strcasecmp()</u>	Compares two strings (case-insensitive)	3
<u>strchr()</u>	Finds the first occurrence of a string inside another string (alias of strpos())	3
<u>strcmp()</u>	Compares two strings (case-sensitive)	3
<u>strcoll()</u>	Locale based string comparison	4
<u>strcspn()</u>	Returns the number of characters found in a string before any part of some specified characters are found	3
<u>strip_tags()</u>	Strips HTML and PHP tags from a string	3
<u>stripslashes()</u>	Unquotes a string quoted with addslashes()	4
<u>stripos()</u>	Returns the position of the first occurrence of a string inside another string (case-insensitive)	5
<u>stristr()</u>	Finds the first occurrence of a string inside another string (case-insensitive)	3
<u>strlen()</u>	Returns the length of a string	3
<u>strnatcasecmp()</u>	Compares two strings using a "natural order" algorithm (case-insensitive)	4
<u>strnatcmp()</u>	Compares two strings using a "natural order" algorithm (case-sensitive)	4
<u>strncasecmp()</u>	String comparison of the first n characters (case-insensitive)	4
<u>strncmp()</u>	String comparison of the first n characters (case-sensitive)	4
<u>strpbrk()</u>	Searches a string for any of a set of characters	5
<u>strpos()</u>	Returns the position of the first occurrence of a string inside another string (case-sensitive)	3
<u>strrchr()</u>	Finds the last occurrence of a string inside another string	3
<u>strrev()</u>	Reverses a string	3
<u>stripos()</u>	Finds the position of the last occurrence of a string inside another string (case-insensitive)	5
<u>strrpos()</u>	Finds the position of the last occurrence of a string inside another string (case-sensitive)	3
<u>strspn()</u>	Returns the number of characters found in a string that contains only characters from a	3



	specified charlist	
<u>strstr()</u>	Finds the first occurrence of a string inside another string (case-sensitive)	3
<u>strtok()</u>	Splits a string into smaller strings	3
<u>strtolower()</u>	Converts a string to lowercase letters	3
<u>strtoupper()</u>	Converts a string to uppercase letters	3
<u>strtr()</u>	Translates certain characters in a string	3
<u>substr()</u>	Returns a part of a string	3
<u>substr_compare()</u>	Compares two strings from a specified start position (binary safe and optionally case-sensitive)	5
<u>substr_count()</u>	Counts the number of times a substring occurs in a string	4
<u>substr_replace()</u>	Replaces a part of a string with another string	4
<u>trim()</u>	Strips whitespace from both sides of a string	3
<u>ucfirst()</u>	Converts the first character of a string to uppercase	3
<u>ucwords()</u>	Converts the first character of each word in a string to uppercase	3
<u>vfprintf()</u>	Writes a formatted string to a specified output stream	5
<u>vprintf()</u>	Outputs a formatted string	4
<u>vsprintf()</u>	Writes a formatted string to a variable	4
<u>wordwrap()</u>	Wraps a string to a given number of characters	4

## 4.7 The mail() Function

The mail() function is used to send emails.

Syntax

mail(to,subject,message,headers,parameters)

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

### 4.7.1 PHP Simple Text E-Mail



The simplest way to send an email with PHP is to send a simple text email. This is a simple text email where we define the variables and send a mail:

```
<?php
$to = "someone@someplace.com";
$subject = "Test mail";
$message = "Hello! This is a simple text email message.";
$from = "someoneelse@anotherplace.com";
$headers = "From: $from";

mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

#### 4.7.2 PHP Mail Form

Using PHP you can create a feedback form for your website. In this example it sends a text message to a specified e-mail.

When using HTML forms with PHP, any form element in the HTML form will automatically be available to the PHP script.

This is how this example works:

Check if the email input is set

If it is not set (like when the page is first visited) it will output the HTML mail form

If the email input is set (like after the form is filled out) it will send the mail from the form

When submit is pressed after the form is filled out, the page reloads, sees that the email input is set, and sends the mail.

```
<html>
<body>
<?php
if (isset($_REQUEST['email']))
{
    $email = $_REQUEST['email'] ;
    $subject = $_REQUEST['subject'] ;
    $message = $_REQUEST['message'] ;
    mail( "someone@someplace.com", "Subject: $subject",
    $message, "From: $email" );

    echo "Thank you for using our mail form";
}
else
{
    echo "<form method='post' action='mailform.php'>
    Email: <input name='email' type='text' /><br />
    Subject: <input name='subject' type='text' /><br />
    Message:<br />
    <textarea name='message' rows='15' cols='40'>
    </textarea><br />
```



```

<input type='submit' />
</form>";
}
?>
</body>
</html>

```

### 4.7.3 Requirements

For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file.

The mail functions are part of the PHP core. There is no installation needed to use these functions.

### 4.7.4 Runtime Configuration

The behavior of the mail functions is affected by settings in the php.ini file. Mail configuration options:

Name	Default	Description	Changeable
SMTP	"localhost"	Windows only: The DNS name or IP address of the SMTP server	PHP_INI_ALL
smtp_port	"25"	Windows only: The SMTP port number. Available since PHP 4.3	PHP_INI_ALL
sendmail_from	NULL	Windows only: Specifies the "from" address to be used in email sent from PHP	PHP_INI_ALL
sendmail_path	NULL	Unix systems only: Specifies where the sendmail program can be found (usually /usr/sbin/sendmail or /usr/lib/sendmail)	PHP_INI_SYSTEM

### 4.7.5 PHP Mail Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
ezmlm_hash()	Calculates the hash value needed by the EZMLM mailing list system	3
mail()	Allows you to send emails directly from a script	3

## 4.8 Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests for a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

### 4.8.1 How to Create a Cookie

The `setcookie()` function is used to create cookies.

Note: The `setcookie()` function must appear BEFORE the `<html>` tag.

Syntax

```
setcookie(name, value, expire, path, domain);
```

Example

The following example sets a cookie named "uname" - that expires after ten hours.

```
<?php
setcookie("uname", $name, time()+36000);
?>
<html>
<body>
<p>
A cookie was set on this page! The cookie will be active when
the client has sent the cookie back to the server.
</p>
</body>
</html>
```

### 4.8.2 How to Retrieve a Cookie Value

When a cookie is set, PHP uses the cookie name as a variable.

To access a cookie you just refer to the cookie name as a variable.

Tip: Use the `isset()` function to find out if a cookie has been set.

Example

The following example tests if the `uname` cookie has been set, and prints an appropriate message.

```
<html>
<body>
<?php
if (isset($_COOKIE["uname"]))
echo "Welcome " . $_COOKIE["uname"] . "!<br />";
else
echo "You are not logged in!<br />";
?>
</body>
</html>
```



## CHAPTER FIVE MySQL

### 5.1 What is MySQL?

MySQL is a type of SQL database management featured in the Linux hosting plans. A database is an organized collection of information that a computer uses to select and display data. Databases can help organize and enhance our site content. Sites with dynamic pages and/or shopping cart software often need an underlying database structure.

MySQL is also a popular database server which is available for Linux, FreeBSD and other flavors of UNIX, and also Win32 platforms. MySQL is often used as a database back-end to PHP web applications or CGIs invoked by Perl database modules.

MySQL has highly configurable user/permissions model as well as network access permissions configuration. MySQL is typically accessed via the client software at the command prompt on a Linux or FreeBSD server. MySQL is much faster than Oracle or Microsoft Access, it's also considered to be the fastest data base server available.

We can also define MySQL as a relational database management system, which means it stores data in separate tables rather than putting all the data in one big area. This adds flexibility, as well as speed. The SQL part of MySQL stands for "Structured Query Language," which is the most common language used to access databases.

MySQL database server is the most popular open source database in the world. It is extremely fast and easy to customize, due to its architecture. Extensive reuse of code within the software, along with a minimalist approach to producing features with lots of functionality, gives MySQL unmatched speed, compactness, stability, and ease of deployment. Their unique separation of the core server from the storage engine makes it possible to run with very strict control, or with ultra fast disk access, whichever is more appropriate for the situation.

### 5.2 Database Construction

MySQL databases have a standard setup. They are made up of a database, in which is contained tables. Each of these tables is quite separate and can have different fields etc. even though it is part of one database. Each table contains records which are made up of fields.

#### 5.2.1 Databases And Logins

The process of setting up a MySQL database varies from host to host, you will however end up with a database name, a user name and a password. This information will be required to log in to the database.

If you have PHPMysqlAdmin (or a similar program) installed you can just go to it to log in with your user name and password. If not you must do all your database administration using PHP scripts.

## 5.3 Creating A Table

Before you can do anything with your database, you must create a table. A table is a section of the database for storing related information. In a table you will set up the different fields which will be used in that table. Because of this construction, nearly all of a site's database needs can be satisfied using just one database.

Creating a table in PHPMyAdmin is simple, just type the name, select the number of fields and click the button. You will then be taken to a setup screen where you must create the fields for the database. If you are using a PHP script to create your database, the whole creation and setup will be done in one command.

### 5.3.1 Fields

There are a wide variety of fields and attributes available in MySQL and I will cover a few of these here:

Field Type	Description
TINYINT	Small Integer Number
SMALLINT	Small Integer Number
MEDIUMINT	Integer Number
INT	Integer Number
VARCHAR	Text (maximum 256 characters)
TEXT	Text

These are just a few of the fields which are available. A search on the internet will provide lists of all the field types allowed.

### 5.3.2 Creating A Table With PHP

To create a table in PHP is slightly more difficult than with MySQL. It takes the following format:

```
CREATE TABLE tablename {  
  
Fields  
  
}
```

The fields are defined as follows:

fieldname type(length) extra info,

The final field entered should not have a comma after it.

I will give full an example of using these later in the section.



### 5.3.3 The Contacts Database

The contacts database will contain all the contact information for the people you enter and the information will be able to be edited and viewed on the internet. The following fields will be used in the database:

Name	Type	Length	Description
id	INT	6	A unique identifier for each record
first	VARCHAR	15	The person's first name
last	VARCHAR	15	The person's last name
phone	VARCHAR	20	The person's phone number
mobile	VARCHAR	20	The person's mobile number
fax	VARCHAR	20	The person's fax number
email	VARCHAR	30	The person's e-mail address
web	VARCHAR	30	The person's web address

You may be wondering why I have used VARCHAR fields for the phone/fax numbers even though they are made up of digits. You could use INT fields but I prefer to use VARCHAR as it will allow dashes and spaces in the number, as well as textual numbers (like 1800-COMPANY) and as we will not be initiating phone calls from the web it is not a problem.

There is one other thing you should be aware of in this database. The id field will also be set as PRIMARY, INDEX, UNIQUE and will be set to auto\_increment (found under Extra in PH

PMYAdmin). The reason for this is that this will be the field identifier (primary and index) and so must be unique. The auto increment setting means that whenever you add a record, as long as you don't specify an id, it will be given the next number.

If you are using PHPMYAdmin or a management program you can now create this in a table called contacts.

## 5.4 Connect to MySQL Server

The first thing you'll want to do is to establish a connection to your MySQL database server. This is accomplished with the `mysql_connect` function. Here's an example:

```
$connid = mysql_connect('servername', 'username', 'password');
```

`$connid` is an identifier (a positive integer, returned by the `mysql_connect` function) that identifies this connection and is used in subsequent function calls to tie those calls to this connection. The `mysql_connect` function is provided three parameters, the name of the server (commonly 'localhost' when the server is on the same computer as PHP), the username and password authorized to access the server.

Next, we need to check that the connection was successfully established. For the sake of this example we are going to print a message on the screen to let us know of whether



or not the call was good. In practice, you will probably take a different course of action (see also the "error control operator, below.)

```
if ($connid == false)
{ print "Server connection failed"; }
else
{ print "Server connected!"; }
```

When we are all done, it is a sound programming practice to close things that we have opened. To close our connection we will use the `mysql_close` function. Notice that we use the `$connid` to reference the connection that we opened.

```
mysql_close ($connid);
Nice and tidy!
```

#### 5.4.1 Using The Error Control Operator

The error control operator can be used to prevent PHP from displaying an error message in the user's browser when the attempt to connect to the MySQL server fails. In the real world, this is the more common requirement. The error control operator is an `@` sign placed next to the function call like this:

```
$connid = @mysql_connect ('servername' , 'username' , 'password');
```

#### 5.4.2 Using the Die Function

Another common real world practice is to use the `die` function to terminate the program and display a message. Without the connection to the database, it's most likely that there's no point in continuing!

```
$connid = @mysql_connect ('servername' , 'username' , 'password') or die("Connection to the DBMS server failed");
```

### 5.5 Creating a Database

Create a database in MySQL with PHP.

MySQL Syntax

```
CREATE DATABASE database_name
```

Now we use this together with the `mysql_query()` function.

All we have to do is to add the MySQL syntax to the `mysql_query()` function.

Example

Here we create a database called "my\_db":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
$sql = "CREATE DATABASE my_db";
if (mysql_query($sql,$con))
{
    echo "Database my_db created";
}
else
```



```

{
    echo "Error creating database: " . mysql_error();
}
?>

```

MySQL Syntax To create a table in a database:

```

CREATE TABLE table_name
(
    column_name1 data_type,
    column_name2 data_type,
    .....
)

```

Now we use this together with the `mysql_query()` function.

Example

This example demonstrates how you can create a table named "Person", with three columns. The column names will be "FirstName", "LastName" and "Age":

```

mysql_select_db("my_db", $connection);
$sql = "CREATE TABLE Person
(
    FirstName varchar(15),
    LastName varchar(15),
    Age int
)";
mysql_query($sql,$con);

```

Note: A database must be selected before a table can be created. This is done in the first line of the example above.

Note: While using PHP to create the varchar data type in a table, you must add the max length parameter, like shown above.

Here is the different MySQL data types that can be used:

Numbers	Description
int(size)	Hold integers only. The maximum number of digits are specified in parenthesis
smallint(size)	
tinyint(size)	
mediumint(size)	
bigint(size)	
decimal(size,d)	Hold numbers with fractions. The maximum number of digits are specified in "size". The maximum number of digits to the right of the decimal is specified in "d"
double(size,d)	
float(size,d)	

Text	Description
char(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis
varchar(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis
tinytext	Holds a variable string with a maximum length of 255 characters
text	Holds a variable string with a maximum length of 65535



blob	characters
mediumtext	Holds a variable string with a maximum length of 16777215 characters
mediumblob	
longtext	Holds a variable string with a maximum length of 4294967295 characters
longblob	

Date	Description
date(yyyy-mm-dd)	Holds date and/or time
datetime(yyyy-mm-dd hh:mm:ss)	
timestamp(yyyymmddhhmmss)	
time(hh:mm:ss)	

Misc	Description
enum(value1,value2,ect)	ENUM is short for ENUMERATED list. Can store one of up to 65535 values listed within the ( ) brackets. If a value is inserted that is not in the list, a blank value will be inserted
set	SET is similar to ENUM. However, SET can have up to 64 list items and can store more than one choice

## 5.6 Primary Key and Auto Increment

Each table should have an unique identifier field. This field is called a primary key.

The primary key field is often an ID number, and is often used with the AUTO\_INCREMENT setting.

When used, AUTO\_INCREMENT adds 1 to the value of the field each time a new entry is added.

To make sure that no primary key fields can be NULL, we add the NOT NULL setting to ensure that the ID value can not be NULL.

Example

This is the same example from above, but with an primary key ID column using AUTO\_INCREMENT and NOT NULL:

```
$sql = "CREATE TABLE Person
(
id int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(id),
FirstName varchar(15),
LastName varchar(15),
Age int
)";
mysql_query($sql,$con);
```

## 5.7 PHP MySQL Functions

Function	Description	PHP
----------	-------------	-----



<u>mysql_affected_rows</u>	Returns the number of affected rows in the previous MySQL operation	3
<u>mysql_change_user</u>	Deprecated. Changes the user of the current MySQL connection	3
<u>mysql_client_encoding</u>	Returns the name of the character set for the current connection	4
<u>mysql_close</u>	Closes a non-persistent MySQL connection	3
<u>mysql_connect</u>	Opens a non-persistent MySQL connection	3
<u>mysql_create_db</u>	Deprecated. Creates a new MySQL database. Use mysql_query() instead	3
<u>mysql_data_seek</u>	Moves the record pointer	3
<u>mysql_db_name</u>	Returns a database name from a call to mysql_list_dbs()	3
<u>mysql_db_query</u>	Deprecated. Sends a MySQL query. Use mysql_select_db() and mysql_query() instead	3
<u>mysql_drop_db</u>	Deprecated. Deletes a MySQL database. Use mysql_query() instead	3
<u>mysql_errno</u>	Returns the error number of the last MySQL operation	3
<u>mysql_error</u>	Returns the error description of the last MySQL operation	3
<u>mysql_escape_string</u>	Deprecated. Escapes a string for use in a mysql_query. Use mysql_real_escape_string() instead	4
<u>mysql_fetch_array</u>	Returns a row from a recordset as an associative array and/or a numeric array	3
<u>mysql_fetch_assoc</u>	Returns a row from a recordset as an associative array	4
<u>mysql_fetch_field</u>	Returns column info from a recordset as an object	3
<u>mysql_fetch_lengths</u>	Returns the length of the contents of each field in a result row	3
<u>mysql_fetch_object</u>	Returns a row from a recordset as an object	3
<u>mysql_fetch_row</u>	Returns a row from a recordset as a numeric array	3
<u>mysql_field_flags</u>	Returns the flags associated with a field in a recordset	3
<u>mysql_field_len</u>	Returns the maximum length of a field in a recordset	3
<u>mysql_field_name</u>	Returns the name of a field in a recordset	3
<u>mysql_field_seek</u>	Moves the result pointer to a specified field	3
<u>mysql_field_table</u>	Returns the name of the table the specified field is in	3
<u>mysql_field_type</u>	Returns the type of a field in a recordset	3
<u>mysql_free_result</u>	Free result memory	3
<u>mysql_get_client_info</u>	Returns MySQL client info	4
<u>mysql_get_host_info</u>	Returns MySQL host info	4
<u>mysql_get_proto_info</u>	Returns MySQL protocol info	4
<u>mysql_get_server_info</u>	Returns MySQL server info	4
<u>mysql_info</u>	Returns information about the last query	4
<u>mysql_insert_id</u>	Returns the AUTO_INCREMENT ID generated from the previous INSERT operation	3
<u>mysql_list_dbs</u>	Lists available databases on a MySQL server	3
<u>mysql_list_fields</u>	Deprecated. Lists MySQL table fields. Use mysql_query() instead	3



<u>mysql_list_processes</u>	Lists MySQL processes	4
<u>mysql_list_tables</u>	Deprecated. Lists tables in a MySQL database. Use <code>mysql_query()</code> instead	3
<u>mysql_num_fields</u>	Returns the number of fields in a recordset	3
<u>mysql_num_rows</u>	Returns the number of rows in a recordset	3
<u>mysql_pconnect</u>	Opens a persistent MySQL connection	3
<u>mysql_ping</u>	Pings a server connection or reconnects if there is no connection	4
<u>mysql_query</u>	Executes a query on a MySQL database	3
<u>mysql_real_escape_string</u>	Escapes a string for use in SQL statements	4
<u>mysql_result</u>	Returns the value of a field in a recordset	3
<u>mysql_select_db</u>	Sets the active MySQL database	3
<u>mysql_stat</u>	Returns the current system status of the MySQL server	4
<u>mysql_tablename</u>	Deprecated. Returns the table name of field. Use <code>mysql_query()</code> instead	3
<u>mysql_thread_id</u>	Returns the current thread ID	4
<u>mysql_unbuffered_query</u>	Executes a query on a MySQL database (without fetching / buffering the result)	4

### 5.7.1 Selecting a Database

Here's something that's not too complicated: to select a MySQL database in PHP use the `mysql_select_db ( )` function, like this:

```
mysql_select_db ("mydatabase");
```

Couldn't be simpler, as long as everything went well -- more on that in a minute.

### 5.7.2 Handling Errors

In the last part of this tutorial series, I discussed the use of the Error Control Operator and the `die ( )` function. Now we'll add one more function to our arsenal. The MySQL support in PHP includes a function called `mysql_error ( )` whose very useful purpose is to provide a text version of an error returned by the MySQL server. We can certainly take advantage of that in our error handling!

I also want to introduce another element here. This is purely a matter of coding technique. There are many ways to write this piece of code, but I feel this form to be quite elegant. You will remember that an "if" statement has this basic format:

```
if (condition)
{do this if condition is true}
else
{do this if condition is false}
```

We can take advantage of that. When a function performs properly it returns with a condition of "true", and when something goes wrong it returns with a "false" condition. Of course, it can also return values, error codes and the like, but for the moment let's ponder just the condition. Since it returns a condition, we can use the function call as the "condition" in an "if" statement. This will allow us to put code in the "true" side of

the statement to do what we want when everything is ok, and to put code in the "false" side to handle our errors.

Let's build an example. First, we're going to connect to the server, and if everything is good, we'll create a database. If there were errors, we'll terminate the program, displaying an error message. In the process of creating the database, if everything goes well, we'll select the database so that we can use it, and if there are errors we'll terminate the program, displaying an appropriate message. Similarly, when selecting the database, we'll use messages to let us know how things went.

Programmatically, this means putting an "if" statement inside the "do this if condition is true" code. When an "if" statement is put into either side of another "if" statement in this fashion, it is known as "nesting" the "if" statements. When you write nested "if"s it is highly recommended that you use indentation and spacing to help you keep track of where you are. It is altogether too easy to have an extra or a missing parenthesis and completely change the logic flow of your program -- often without causing any PHP syntax error. Be warned! Be careful!

Now, here's our code (you might want to spend a little time following through this to make sure you fully understand it):

```
if ( $connid = mysql_connect ( 'localhost' 'username' 'password' ) )
{ print '<p> Connected to MySQL Server. </p>';
  if ( @mysql_query ( 'CREATE DATABASE murphy' ) )
  { print '<p> Database murphy created. <p>';
    if ( @mysql_select_db ( 'murphy' ) )
    { print '<p> Database murphy selected </p>';
    }
    else
    { die ( ' <p> Database selection failed because ' , mysql_error ( ) , ' </p>' )
    }
  }
  else
  { die ( ' <p> Database creation failed because ' , mysql_error ( ) , ' </p>' )
  }
}
else
{ die ( ' <p> Database Server connection failed because ' , mysql_error ( ) , ' </p>' )
}
```

## 5.8 Inserting Data From a Form to a Database

In PHP we can allow the users to use a form to insert or edit data in a database. First we create a form:

```
<form action="insert_db.php" method="POST">
Enter your Firstname: <input type="text" name="firstname" />
Enter your Lastname: <input type="text" name="lastname" />
Enter your Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

Then the "insert\_db.php" page:



```

$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$sql="INSERT INTO person
(firstname,lastname,age)
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
if (!mysql_query($sql,$con))
{
    die('Error: ' . mysql_error());
}
echo "Success!";

```

## 5.9 ODBC With PHP

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database).

### Create an ODBC Connection

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is how to create an ODBC connection to a MS Access Database:

Open the Administrative Tools icon in your Control Panel.

Double-click on the Data Sources (ODBC) icon inside.

Choose the System DSN tab.

Click on Add in the System DSN tab.

Select the Microsoft Access Driver. Click Finish.

In the next screen, click Select to locate the database.

Give the database a Data Source Name (DSN).

Click OK.

Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to set up a DSN for you to use.

### 5.9.1 Connecting to an ODBC

The `odbc_connect()` function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

The `odbc_exec()` function is used to execute an SQL statement.

Example

The following example creates a connection to a DSN called northwind, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('northwind','');  
$sql="SELECT * FROM customers";  
$rs=odbc_exec($conn,$sql);
```

### 5.9.2 Retrieving Records

The `odbc_fetch_rows()` function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false. The function takes two parameters: the ODBC result identifier and an optional row number:

```
odbc_fetch_row($rs)
```

#### Retrieving Fields from a Record

The `odbc_result()` function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.

The code line below returns the value of the first field from the record:

```
$compname=odbc_result($rs,1);
```

The code line below returns the value of a field called "CompanyName":

```
$compname=odbc_result($rs,"CompanyName");
```

### 5.9.3 Closing an ODBC Connection

The `odbc_close()` function is used to close an ODBC connection.

```
odbc_close($conn);
```

#### An ODBC Example

The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
<html>  
<body>  
<?php  
$conn=odbc_connect('northwind','');  
if (!$conn)  
{exit("Connection Failed: " . $conn);}  
$sql="SELECT * FROM customers";  
$rs=odbc_exec($conn,$sql);  
if (!$rs)  
{exit("Error in SQL");}  
echo "<table><tr>";  
echo "<th>Companyname</th>";  
echo "<th>Contactname</th></tr>";  
while (odbc_fetch_row($rs))  
{
```



```
$compname=odbc_result($rs,"CompanyName");  
$conname=odbc_result($rs,"ContactName");  
echo "<tr><td>$compname</td>";  
echo "<td>$conname</td></tr>";  
}  
odbc_close($conn);  
echo "</table>";  
?>  
</body>  
</html>
```

## CHAPTER SIX : PROJECT DESCRIPTION

### 6.1 Introduction

This chapter reflects main content and aims of this Project. This includes usage of symbols and meaning of terms.

This proje formed in three parts. The first is customer part, second is Branch part and last one is Administrative Center part.

This project shortly, creates opportunity to rent car, no matter where you are in the world.

### 6.2 Customer Interface

At customer part, there is product search and a simple shopping process. At this part customer can search any product by name and if exist product, they can buy it. At this part, there are just two payment ways; one of them is bank transfer and another is payment by Credit Card. At this project, the payment processes are not care because we care about processes after payment.

When we enter the system we will see a page like follow figure 2.1

The screenshot shows a web browser window with the URL 'www.tekno-city.com' in the address bar. The page has a green header bar with the website name. Below the header, there is a blue sidebar on the left and a main blue content area on the right. The sidebar contains a 'Home Page' link, a 'Customer Login' section with input fields for 'ID number' and 'Password', a 'Login' button, and a 'Create New User' link. The main content area features a 'Product Name' input field and a 'Search' button.

Figure 2.1

At the left part of page, there are links to return homepage and to create new customer. Also at this side, there is login form to customer enter as a member. At right part, there is a search form for customer to search any product. When customer search any product, system will go to search page and shows the result whether find product or not. If there is not any product system shows a message as in figure 2.2





**Figure 2.2**

If there is a product which customer searched, system will show product information as in figure 2.3



**Figure 2.3**

As shown at figure, customer just can see information but can not buy it. To do this process, they must login to system. They will enter ID number an password into login form at left side as shown in figure 2.4



**Figure 2.4**

After logged into system, customer can buy product by shown a buy button next to information shown as in figure 2.5



**Figure 2.5**

When customer click to “buy” button, selling process will be done.





**Figure 2.6**

### 6.3 Branch Interface

At this part of project there are branch processes. At this side, when center sent product these have a certain value, will register into system, selling, updating, product distribution in the store (sell department, service department, service car, and marketing...) , manager hierachical in branch and other process performing. At this side, the most important part is managers control permissions of program. At system there are three administration level.. Low level (i.e secreter), middle level(i.e manager) and top level is administrator. From low to top there is permission diffrentions. At low level admin can do just, enter products to system and updating them. At figure 3.1 shown.



**Figure 3.1**

Another admin level is manager level. At this level, manager has more permission than secreter. Login page shown in figure 3.2

Username:	<input type="text" value="manager"/>
Password :	<input type="password" value="....."/>
City	<input type="text" value="Lefkoşa"/> <input type="button" value="v"/>
<input type="button" value="Login"/>	

**Figure 3.2**

At figure 3.3 looking manager page.





**Figure 3.3**

Manager's extra permissions are deleting stocks (i.e selling), creating new product catalog.

The top level of permission is administrator. The administrator has full control of system. In figure 3.4 shows administrator page



**Figure 3.4**

As shown in above figure, the administrator has some more important tasks. These are, creating administrator, show stocks and show report.

### 6.3.1 Branch Processes

The processes perform in branch as at below:

**Add Category:** At this part, can creating new product category. At figure 3.5 , showing creating a new category as “modem”

The screenshot shows the 'Add Category' form on the website. The header is green with the text 'www.tekno-city.com' in red. On the left, there is a vertical menu with buttons: 'Home Page', 'Show Report', 'Administrator', 'Show Stock', 'Add Category', 'Delete Stock', and 'Add Stock'. The main content area is blue. It contains a label 'Category Name:' followed by a text input field containing 'Modem'. Below this is a button labeled 'Add Category'.

Figure 3.5

**Add Stock:** At this part, the products register to systems. Shown at figure 3.6

The screenshot shows the 'Add Stock' form on the website. The header is green with the text 'www.tekno-city.com' in red. On the left, there is a vertical menu with buttons: 'Home Page', 'Show Report', 'Administrator', 'Show Stock', 'Add Category', 'Delete Stock', and 'Add Stock'. The main content area is blue. It contains several labels and input fields: 'Category' with a dropdown menu showing 'Dvd Writer', 'Stock Name' with a text input field containing 'Lg', 'Quantity' with a text input field containing '12', 'Serial no' with a text input field containing '545616546546514', 'Buy from' with a text input field containing 'Teknolojix Ltd..', 'Price' with a text input field containing '80', and 'Location in Store' with a text input field containing 'Store'. Below these fields is a button labeled 'Add'.

Figure 3.6

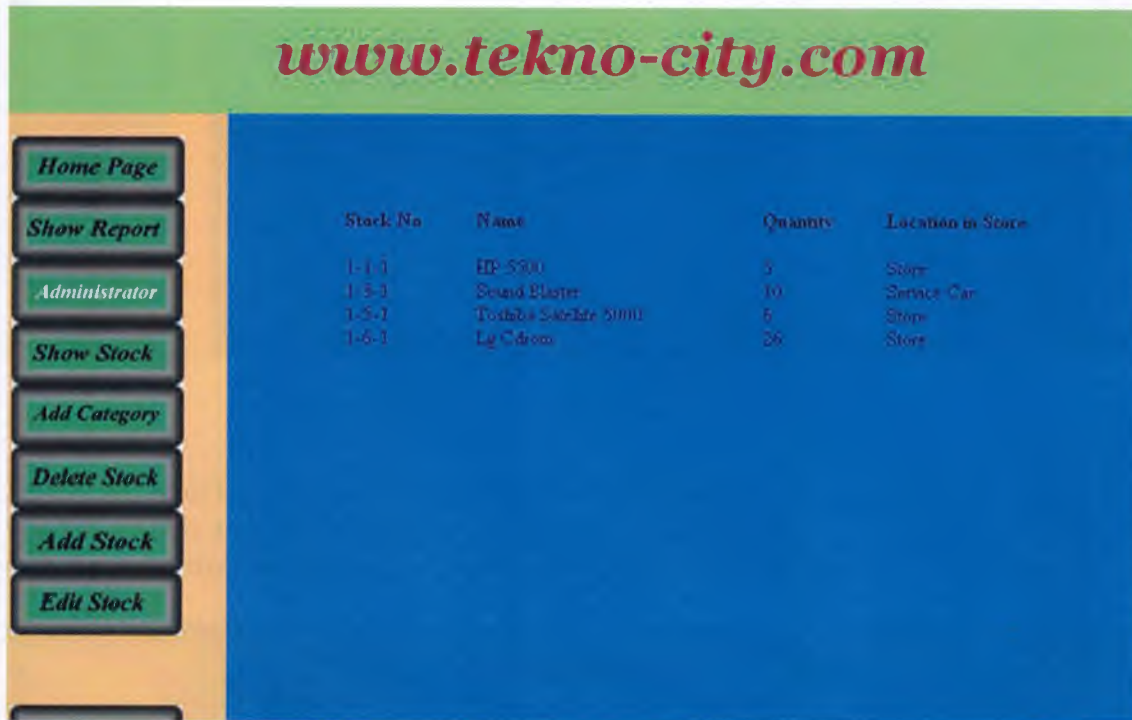
The information can be expressed such as: Category is, the stock category which included in. Stock Name is the name of product. Quantity is how many product at this type. Serial no is the the unique number of product. It is using especially for guaranty and other following process. Buying from is also using for service condition which is shows us where is this product buying. Price is the sell price of product. Location in



store shows that where the product located. It can be in store, in selling department, marketing department or service department.

**Show Report:** It is shows a brief report about product informations (quantity, price, total value) about products at branch.

**Show Stock:** This part shows the informations at the hand. How many items,their number, locations and other informations. Showing in figure 3.7



Stock No	Name	Quantity	Location in Store
1-1-1	HP-5500	3	Store
1-3-1	Sound Blaster	10	Service Car
1-5-1	Toshiba Satellite 5000	5	Store
1-6-1	Lg Cdrum	26	Store

Figure 3.7

**Edit Stock:** Sometimes, some products information can be changed. Since to more reasons, for example, crashes, mistakes, going to service and return, going to marketing and return, selling and so reasons can effects the stock. The major changes are, quantity changes, price value changes. We can give an easy example as: when a prôduct needs to change some part of it, we change in guaranty time, so the changed aprt not sell. Here, there is just a quantity changing. Selling is change quantity and price value.

Update form shown in figure 3.8



**Figure 3.8**

Firstly, we search product will be change. We enter the stock number. If there is no product system will give a warning that it doesn't find product. When system find the product we see a page as in figure 3.9



**Figure 3.9**

As shown in figure, we changed just how many items (quantity), and edit reason. Stock number, serial no and other informations can not be changed. Edit reasons are “go oto service”, “come from service”, “selling”, “go to marketing”, “come from marketing”,



and other process up to company work. Respect to reason, system will change informations (price, quantity and others).

**Administrator management:** It is using to creating admin in branch. Only top level administrator perform that. In a branch there is just only one top level administrator and can not create more. Administrator can create low level or middle level admin. Create admin form shown in figure 3.10



The screenshot shows a web application interface for 'www.tekno-city.com'. The main content area is blue, and the header is green. On the left, there is a vertical menu with buttons: 'Home Page', 'Show Report', 'Administrator', 'Show Stock', 'Add Category', 'Delete Stock', 'Add Stock', and 'Edit Stock'. The main area contains a form for creating a new administrator. The form has the following fields: 'User Name', 'Password', 'Admin Level' (a dropdown menu with 'Select a level' as the selected option), 'Name', 'Surname', and 'Mail'. A 'Create New Admin' button is located below the form fields.

Figure 3.10

## 6.4 Administrative Center Interface

It is the main control center of branches. Here is headquarters of company. At this side, reporting of branches showing. When a manager at Center, enter to system they will see the list of branches. By selecting a branch, they go to report page. In this page can see a short report and can download full information report by click "download report" link.

In figure 4.1 shows login page for Center.

<b>Username:</b>	<input type="text" value="nicosia_admin"/>
<b>Password :</b>	<input type="password" value="....."/>
<input type="button" value="Login"/>	

**Figure 4.1**

After login to system, homepage of center will be seen. In the page there is a list of cities. By selecting a city then click the show report button. Shown in figure 4.2



**Figure 4.2**

At report page there is summarize of report about that branch. Solded, changed and reamin stock in hand values are shown. For extra information (full report), can be clicking "download as file" link. In figure 4.3 shown as:





Figure 4.3

After clicking link, a download query message will appear. By clicking save can save or directly can open. Shown in figure 4.4



Figure 4.4

## CHAPTER SEVEN: PROJECT APPLICATION CODES

At this chapter there are some important project's file source codes

### 7.1 Customer Interface Codes

#### 7.1.1 index.php

```
<?
session_start();
require_once("mahsum_db.php");
require_once("customer_function.php");
$connection=mahsum_db_baglanti();
$sess=session_id();
make_session($sess);

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">

<script type="text/javascript" language="javascript">
<!--
function validator(form)
{

    if (form.customer_id.value == "")
    {
        alert("Please enter your ID Number and try again!...");
        form.customer_id.focus();
        return (false);
    }

    if (form.passwrđ.value == "")
    {
        alert("Please enter your password and try again!...");
        form.passwrđ.focus();
        return (false);
    }

    return (true);
}

function search_validator(form1)
{
```



```

if (form1.stock_name.value == "")
{
    alert("Please enter a stock name and try again!...");
    form1.stock_name.focus();
    return (false);
}
return (true);
}

//-->
</script>
</head>

<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">
<table width="100%" height="597" border="0" cellpadding="0" cellspacing="0">
<tr align="center" valign="middle" bgcolor="#A2F09F">
<td height="97" colspan="3"><strong><font color="#AC1A58" size="+7"
face="Georgia, Times New Roman, Times, serif"><em>www.tekno-
city.com</em></font></strong></td>
</tr>
<tr align="left" valign="top">
<td width="21%" rowspan="2" bgcolor="#EFBF8F"> <br>
<? require_once("menu.php"); ?>
<br>
</td>
<td height="479" colspan="2"><table width="100%" border="0" cellspacing="0"
cellpadding="0">
<tr align="left" valign="top">
<td width="2%" height="479">&nbsp;&nbsp;&nbsp;</td>
<td width="98%"><br> <br> <br> <br>
<?
if(!$op)
{
?>
<form name="form1" method="post" action="index.php?op=1"
onsubmit="return search_validator(this)">
<p>&nbsp;&nbsp;&nbsp;</p>
<p>&nbsp;&nbsp;&nbsp;</p>
<p>&nbsp;&nbsp;&nbsp;</p>
<p>&nbsp;&nbsp;&nbsp;</p>
<table width="100%" border="0" cellspacing="0" cellpadding="0"
height="114">
<tr>
<td width="47%" height="1" align="right">Product
Name&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
<td width="53%" height="1"> <input name="stock_name" type="text"
id="stock_name" maxlength="10" size="20"></td>
</tr>
<tr align="center">

```

```

        <td colspan="2" height="26"> <input type="submit" name="Submit"
value=" Search " >
        </td>
    </tr>
</table>
</form>
<?
}
else if($op==1)
{

// $stock_add_date = time();

$get_stock=mysql_query("select * from stock where stock_name = '$stock_name' ");
$stock_number=mysql_num_rows($get_stock);

if($stock_number<1)
{
echo "<center><b>there is no any product you searched</center></b>";
echo "<br>";
}
else
{
?>
        <br> <br> <br>
        <table width="79%" border="0" cellspacing="0" cellpadding="0">
        <tr align="left" valign="top">
            <td width="21%"> <p><font color="#FFFFFF"><strong>Stock
No</strong></font></p></td>
            <td width="35%"><font color="#FFFFFF"><strong>Stock
Name</strong></font></td>
            <td width="15%"><font
color="#FFFFFF"><strong>Price</strong></font></td>
            <td width="29%">&nbsp;</td>
        </tr>
        <tr align="left" valign="top">
            <td>&nbsp;&nbsp;&nbsp;</td>
            <td>&nbsp;&nbsp;&nbsp;</td>
            <td>&nbsp;&nbsp;&nbsp;</td>
            <td>&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <?
        for($i=0; $i<$stock_number; $i++)
        {
$stocks=mysql_fetch_array($get_stock);
$s_name = $stocks['stock_name'];
$s_no = $stocks[stock_no];
$s_price = $stocks[price];
$s_quantity = $stocks[quantity];

```





### 7.1.2 mahsum\_db.php

<?

```
function mahsum_db_baglanti()
{
    $result = mysql_pconnect("localhost", "mahsum", "123456");
    if (!$result)
        return false;
    if (!mysql_select_db("project"))
        return false;

    return $result;
}

?>
```

### 7.1.3 transaction.php

<?

```
session_start();
require_once("mahsum_db.php");
require_once("customer_function.php");
$connection=mahsum_db_baglanti();
```

?>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>
```

```
<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">
<table width="100%" height="597" border="0" cellpadding="0" cellspacing="0">
  <tr align="center" valign="middle" bgcolor="#A2F09F">
    <td height="97" colspan="3"><strong><font color="#AC1A58" size="+7"
face="Georgia, Times New Roman, Times, serif"><em>www.tekno-
city.com</em></font></strong></td>
  </tr>
  <tr align="left" valign="top">
    <td width="21%" rowspan="2" bgcolor="#EFBF8F"> <br>
      <? require_once("menu.php"); ?>
      <br>
    </td>
```



```

<td height="479" colspan="2"><table width="100%" border="0" cellspacing="0"
cellpadding="0">
  <tr align="left" valign="top">
    <td width="2%" height="479">&nbsp;</td>
    <td width="98%"><br>
      <?
$stock_sell_date = time();
$stock_no=$s_id;

$get_stock_tr=mysql_query("select * from stock where stock_no='$s_id'");

$stocks=mysql_fetch_array($get_stock_tr);
$s_name = $stocks['stock_name'];
$s_price = $stocks[price];
$s_quantity = $stocks[quantity];
$s_city_no = $stocks[city_no];
if($s_quantity<1)
{
echo "we do not have this product now in stocks";
}
else
{
$sell_stock=mysql_query("insert into transaction values
(NULL,'$idnumber','$stock_no',1,$s_price,$s_city_no,$stock_sell_date)");
echo "<br>"
      . "<br>"
      . "<br><br>"
      . "<strong>Name: $s_name </strong> <br>"
      . "<strong>Quantity :1</strong><br>"
      . "<strong>Total Price: $s_price $</strong>";

$upd_stock=mysql_query("update stock set quantity=$s_quantity-1 where
stock_no='$s_id'");
}

?>
  </td>
</tr>
</table> </td>
</tr>
<tr>
  <td width="45%" height="19" align="center" valign="top"></td>
  <td width="34%" align="center" valign="top">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

#### 7.1.4 customer\_function.php

```

<?
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();

function customer_login($customer_id, $passwrđ)
{

    $customer_login = mysql_query("select * from customer where
customer_id='$customer_id' and passwrđ='$passwrđ'");
    if (!$customer_login)
        return false;

    if (mysql_num_rows($customer_login)>0)
        return true;
    else
        return false;
}

//-----
function check_customer()
{
    global $idnumber;
    if (!session_is_registered("idnumber"))
    {
        echo "<font size='5' color='red'>Error</font><br>";
        echo "this zone need user login.. please enter your is number and password!.<br>";
        exit;
    }
}

//-----
function make_session($session_id)
{
    if (!$session_id)
        return false;
    $make_sess=mysql_query("insert into customer_session values ('$session_id',0)");
    return true;
}

//-----

function customer_is_login($sess_id)
{
    if (!$sess_id)
        return false;

    $log_user = mysql_query("select * from customer_session where session_id='$sess_id'
AND status=1");
    if (!$log_user)
        return false;
}

```



```

    if (mysql_num_rows($log_user)>0)
        return true;
    else
        return false;
}

//-----
function update_session($session_id)
{
    if (!$session_id)
        return false;
    $update_sess=mysql_query("update customer_session set status=1 where
session_id='$session_id'");
    return true;
}
//-----
function delete_session($session_id)
{
    if (!$session_id)
        return false;
    $delete_sess=mysql_query("delete from customer_session where
session_id='$session_id'");
    return true;
}
//-----

```

?>

## 7.2 Branch Interface Codes

### 7.2.1 index.php

```

<?
session_start();
require_once("admin_function.php");
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();

if ($username && $password && $city_no)
{
    if (admin_login($username, $password, $city_no ))
    {
        $administrator=$username;
        session_register("administrator");
    }
}

```

```

setcookie(cityy_no,$city_no,36000);
}
else
{

    echo "<font color='red'>HATA!!!..</font><br>";
    echo "Yönetici adınızı veya Şifrenizi yanlış girdiniz.Lütfen kontrol edin ve yeniden deneyin";

    exit;
}
}

```

```

check_administrator();

```

```

$permission =get_permission($administrator);
// 1= super admin    2= manager of office    3=normal user in office

```

```

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>

<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">
<table width="100%" height="575" border="0" cellpadding="0" cellspacing="0">
  <tr align="center" valign="middle">
    <td height="97" colspan="3" bgcolor="#A2F09F"><strong><font color="#AC1A58"
size="+7" face="Georgia, Times New Roman, Times, serif"><em>www.tekno-
city.com</em></font></strong></td>
  </tr>
  <tr align="left" valign="top">
    <td width="21%" rowspan="2">
      <? require_once("menu.php"); ?>
    </td>
    <td height="451" colspan="2"><div align="center"><br>
      <br>
      <font size="4"><br>
      <strong>Welcome to Tekno-City Management Department<br>
      <br>

```



```

        You can find whatever you can do at left side menu<br>
        <br>
        All menu are named with their to do job<br>
        <br>
        After finish your work, for your security please use LOG OUT button at
        the bottom ..</strong></font></div></td>
    </tr>
    <tr>
        <td width="45%" height="19" align="center" valign="middle">&nbsp;</td>
        <td width="34%" align="center" valign="top">&nbsp;</td>
    </tr>
</table>
</body>
</html>

```

### 7.2.2 add\_stock.php

```

<?
session_start();
require_once("admin_function.php");
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();
check_administrator();
$permission =get_permission($administrator);
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>

<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">
<table width="100%" height="578" border="0" cellpadding="0" cellspacing="0">
    <tr align="center" valign="middle">
        <td height="97" colspan="3" bgcolor="#A2F09F"><strong><font color="#AC1A58"
size="+7" face="Georgia, Times New Roman, Times, serif"><em>www.tekno-
city.com</em></font></strong></td>
    </tr>
    <tr align="left" valign="top">
        <td width="21%" rowspan="2">
            <? require_once("menu.php"); ?>
        </td>
        <td height="462" colspan="2">
            <?
if(!$op)
{
$get_cat=mysql_query("select * from category "); //show category query
$cat_num=mysql_num_rows($get_cat);

```





```

        <td height="21">Price</td>
        <td height="21" align="left" valign="top">
            <input name="price" type="text" maxlength="50" size="20"></td>
    </tr>
    <tr>
        <td height="21">Location in Store</td>
        <td height="21" align="left" valign="top">
            <input name="loc_in_store" type="text" id="loc_in_store" size="20"
maxlength="50"></td>
    </tr>
    <tr align="center">
        <td colspan="2" height="26"> <input type="submit" name="Submit" value="
Add  ">
    </td>
</tr>
</table>
</form>

    <br>
    <br>
    <br>
    <br>
    <?
    }
    else if($op==1)
    {
        $get_cat_num=mysql_query("select * from stock where cat_no=$cat_no"); //show
category query
        $cat_num_stock=mysql_num_rows($get_cat_num);
        $scat_no=$cat_num_stock+1;

        $stock_add_date = time();

        $city_no=$cityy_no;

        $stock_no=$city_no."-".$cat_no."-".$scat_no;

        $insert_stock=mysql_query("insert into stock values
('$stock_no','$cat_no','$city_no','$stock_name',$quantity,'$serial_no','$buy_from',$price,$
loc_in_store',$stock_add_date)");

        $insert_stock=mysql_query("insert into center_stock values
('$stock_no','$cat_no','$city_no','$stock_name',$quantity,'$serial_no','$buy_from',$price,$
loc_in_store',$stock_add_date)");

        echo "Your stocks inserted to database<br><br><br><br>";
        echo "Please wait";
        echo "<meta http-equiv='refresh' content='3; URL=add_stock.php'>";
    }
}

```

```

}
else
{
echo "";
}
?>

```

```

<div align="center"></div>

```

```

</td>
</tr>
<tr>
<td width="45%" height="19" align="center" valign="top">&nbsp;</td>
<td width="34%" align="center" valign="top">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

### 7.2.3 edit\_stock.php

```

<?
session_start();
require_once("admin_function.php");
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();
check_administrator();
$permission =get_permission($administrator);
?>

```

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>

```

```

<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">
<table width="100%" height="590" border="0" cellpadding="0" cellspacing="0">
<tr align="center" valign="middle">
<td height="97" colspan="3" bgcolor="#A2F09F"><strong><font color="#AC1A58"
size="+7" face="Georgia, Times New Roman, Times, serif"><em>www.tekno-
city.com</em></font></strong></td>
</tr>

```



```

<tr align="left" valign="top">
  <td width="21%" rowspan="2">
    <? require_once("menu.php"); ?>
  </td>
  <td height="469" colspan="2">
    <?
if(!$op)
{
?>
  <form name="form1" method="post" action="edit_stock.php?op=1">
    <p>&nbsp;</p>
    <p>&nbsp;</p>
    <p>&nbsp;</p>
    <p>&nbsp;</p>
    <table width="100%" border="0" cellpadding="0" cellspacing="0" height="114">
      <tr>
        <td width="21%" height="1">Stock No</td>
        <td width="79%" height="1">
          <input name="stock_no" type="text" id="stock_no" maxlength="10"
size="20"></td>
        </tr>

      <tr align="center">
        <td colspan="2" height="26">
          <input type="submit" name="Submit" value="  Search  ">
        </td>
      </tr>
    </table>
  </form>

  <?
}
else if($op==1)
{
$find_stock=mysql_query("select * from stock where stock_no='$stock_no'");
$s_num=mysql_num_rows($find_stock);

if($s_num<1)
{
echo "<br><br><br><br>There is not any product has $stock_no
number<br><br><br>";
echo "<a href='edit_stock.php'>Go Back</a>";
}
else
{
$stck=mysql_fetch_array($find_stock);
$s_name = $stck['stock_name'];
$s_quantity = $stck[quantity];

```

```

$ss_serial_no = $stck['serial_no'];
$ss_buy_from = $stck['buy_from'];
$ss_price = $stck['price'];

```

```

echo "<br><br><br>";
echo "Name : $ss_name <br> Total quantity : $ss_quantity <br>"
." Serial no : $ss_serial_no <br> Buying From : $ss_buy_from"
."";
?>

```

```

<form name="form1" method="post" action="edit_stock.php?op=2">
  <table width="100%" border="0" cellpadding="0" cellspacing="0" height="248">
    <tr>
      <td width="21%" height="29"><strong>Stock No</strong></td>
      <td width="79%" height="29"> <? echo "$stock_no"; ?>
        <input name="stock_no" type="hidden" value="<?=$stock_no; ?>" >
        <input name="stock_name" type="hidden" id="stock_name"
value="<?=$s_name; ?>">
        <input name="serial_no" type="hidden" id="serial_no"
value="<?=$ss_serial_no; ?>">
        <input name="buy_from" type="hidden" id="buy_from"
value="<?=$ss_buy_from; ?>">
        <input name="price" type="hidden" id="price" value="<?=$ss_price; ?>">
        <input name="total_quantity" type="hidden" id="total_quantity"
value="<?=$ss_quantity; ?>"> </tr>
    <tr>
      <td height="20"><strong>Quantity</strong></td>
      <td height="20">
        <input name="quantity" type="text" maxlength="50" size="20"></td>
    </tr>
    <tr>
      <td height="21" align="left" valign="top"><strong>Edit Reason</strong></td>
      <td height="21" align="left" valign="top">
        <select name="edit_reason" id="edit_reason">
          <option disabled>Select a Edit Reason</option>
          <?
            $get_reason=mysql_query("select * from reason");
            $reason_num=mysql_num_rows($get_reason);

            for($rn=0; $rn<$reason_num; $rn++)
            {
              $reas=mysql_fetch_array($get_reason);
              $reason = $reas['reason'];
              echo "<option value='$reason'$reason</option>";
            }
          ?>
        </select>
      </td>
    </tr>
  </table>

```



```

<tr>
  <td height="21">&nbsp;</td>
  <td height="21"><input type="submit" name="Submit2" value="    Update
Stock    ">
</td>
</tr>

<tr align="center">
  <td colspan="2" height="26">&nbsp;</td>
</tr>
</table>
</form>

<?
}
}
else if($op==2)
{
$edit_date=time();

$stock_no=$stock_no;
$stock_serial_no=$serial_no;
$stock_name=$stock_name;
$stock_quantity=$quantity;
$stock_price=$price;
$stock_city_no=$cityy_no;
$stock_buy_from=$buy_from;
$edit_reason=$edit_reason;
$date=$edit_date;

$insert_stock_update=mysql_query("insert into updated values (
NULL,'$stock_no','$stock_serial_no','$stock_name',$stock_quantity,$stock_price,$stoc
k_city_no,$stock_buy_from','$edit_reason',$date)");

if($insert_stock_update)
{
  $get_reason_up=mysql_query("select * from reason where reason='$edit_reason'");
  $reas_up=mysql_fetch_array($get_reason_up);
  //$price_eff=$reas_up[price_effect];
  $quantity_eff=$reas_up[quantity_effect];
  if($quantity_eff == 0)
    $quantity_diff=$total_quantity-$quantity;
  else
    $quantity_diff=$total_quantity+$quantity;

  $update_stock=mysql_query("update stock set quantity=$quantity_diff where
stock_no='$stock_no'");
  if($update_stock)
  {

```

```

echo "<br><br><br><br><center>";
echo "Your product informations updated <br><br><br><br>";
echo "Please wait";
echo "</center>";
echo "<meta http-equiv='refresh' content='5; URL=edit_stock.php'>";
}
}

```

```

}

else
{

echo "Wrong operation...";
echo "<br><br><br>";
}
?>

```

```

<div align="center"></div>

```

```

        </td>
</tr>
<tr>
    <td width="45%" align="center" valign="top">&nbsp;</td>
    <td width="34%" align="center" valign="top">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

#### 7.2.4 admin\_manage.php

```

<?
session_start();
require_once("admin_function.php");
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();
check_administrator();

$permission =get_permission($administrator);
// 1= super admin    2= manager of office    3=normal user in office

```





```

        <option value="2">Manager</option>
        <option value="3">Low Admin</option>
    </select></td>
</tr>
<tr>
    <td><strong>Name</strong></td>
    <td><input name="name" type="text" id="name"></td>
</tr>
<tr>
    <td><strong>Surname</strong></td>
    <td><input name="surname" type="text" id="surname"></td>
</tr>
<tr>
    <td><strong>Mail</strong></td>
    <td><input name="mail" type="text" id="mail"></td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr align="center" valign="top">
    <td colspan="2">
        <input type="submit" name="Submit" value="Create New Admin">
    </td>
</tr>
</table>
</form>
<?
}
else if($op==1)
{
    $date=time();

    $insert_stock=mysql_query("insert into admin values (
NULL,'$admin_name','$admin_psswr','$name','$surname','$admin_level',$cityy_no,'$m
ail',$date)");
    echo "<br><br>Cretaed new administrator";
    echo "<meta http-equiv='refresh' content='3; URL=admin_manage.php'>";
    }
    else
    {
        echo "wrong operation";
        echo "<br>";
    }

?>

```



```

        </td>
    </tr>
    <tr>
        <td width="45%" height="19" align="center" valign="middle">&nbsp;</td>
        <td width="34%" align="center" valign="top">&nbsp;</td>
    </tr>
</table>
</body>
</html>

```

## 7.3 Administrative Center Interface Codes

### 7.3.1 index.php

```

<?
session_start();
require_once("center_function.php");
require_once("mahsum_db.php");
$connection=mahsum_db_baglanti();

if ($administrator && $administrator_psswrđ)
{
    if (center_login($administrator, $administrator_psswrđ))
    {

        $adm=$administrator;
        session_register("adm");
    }
    else
    {

        echo "<font color='red'>HATA!!!</font><br>";
        echo "Yönetici adınızı veya Şifrenizi yanlış girdiniz.Lütfen kontrol edin ve yeniden
        deneyin";
        exit;
    }
}

check_center_adm();

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Stock Control</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</head>

<body bgcolor="#687BFD" leftmargin="1" topmargin="1" marginwidth="0"
marginheight="0">

```





```

        </tr>
        <tr align="center" valign="top">
            <td colspan="2" bgcolor="#D1D1D1"> <input type="submit" name="Submit"
value="    Show Report    "></td>
        </tr>
    </table>
</form>

    <?

    }
    else if($op==1)
    {
        $file_n="city_reports.rtf";
        if(is_file($file_n))
        {
            unlink($file_n);
        }
        @ $fp = fopen($file_n, "a+");

$sum_tr=0; //total transaction value
    $sum_st=0; //total stock value
    $sum_up=0; //total updated price

    $fp_center_wr="Stock Report of city number $city_no\n\n\n\n"; //wr
write
    $fp_center_status="Stock_no\t\t\t Quantity\t\t\t Status\n\n";
        fwrite($fp, $fp_center_wr);
        fwrite($fp,$fp_center_status);

        $cent_stck=mysql_query("select * from center_stock where
city_no=$city_no order by stock_no");
        $cent_stck_num=mysql_num_rows($cent_stck);
        for($csn=1; $csn<=$cent_stck_num; $csn++) // all stocks at center sent
to this branch
        {
            $center=mysql_fetch_array($cent_stck);
            $stock_no = $center['stock_no'];

            $branch_tr=mysql_query("select * from transaction where
city_no=$city_no AND stock_no='$stock_no'");
            $total_tr=mysql_num_rows($branch_tr);
            for($itr=0; $itr<$total_tr; $itr++)
            {
                $tr=mysql_fetch_array($branch_tr);
                $tr_quantity = $tr[quantity];
                $tr_price = $tr[price];
                $tp1=$tr_quantity*$tr_price;
                $sum_tr=$sum_tr+$tp1;
                $report_var_tr="$stock_no\t\t\t\t $tr_quantity\t\t\t\t Sold\n";
            }
        }
    }
}

```

```

        fwrite($fp, $report_var_tr);
    }

    $branch_st=mysql_query("select * from stock where city_no=$city_no
AND stock_no='$stock_no'");
    $total_st=mysql_num_rows($branch_st);

    for($i=0; $i<$total_st; $i++)
    {
        $st=mysql_fetch_array($branch_st);
        $st_quantity = $st[quantity];
        $st_price = $st[price];
        $tp4=$st_quantity*$st_price;
        $sum_st=$sum_st+$tp4;
        $report_var_st="$stock_no\t\t\t\t $st_quantity\t\t\t\t at store\n";
        fwrite($fp, $report_var_st);
    }

    $branch_up=mysql_query("select * from updated where
stock_city_no=$city_no AND stock_no='$stock_no'");
    $total_up=mysql_num_rows($branch_up);

    for($iup=0; $iup<$total_up; $iup++)
    {
        $sup=mysql_fetch_array($branch_up);
        $sup_quantity = $sup[stock_quantity];
        $sup_price = $sup[stock_price];
        $sup_reason = $sup['edit_reason'];

        $tp2=$sup_quantity*$sup_price;
        $sum_up=$sum_up+$tp2;
        $report_var_up="$stock_no\t\t\t\t $sup_quantity\t\t\t\t
$sup_reason\n";
        fwrite($fp, $report_var_up);
    }
}

```



```

$report_total_tr="\n\nTotal value of sold stock is $sum_tr"
."n\n-----\n\n";
    fwrite($fp, $report_total_tr);
    // transaction finish

$sum_st"
    $report_total_st="\n\nTotal value of Stocks in Store is
    ."n\n-----
-----\n\n\n";
    fwrite($fp, $report_total_st);
    // stock finish

stock is $sum_up"
    $report_total_up="\n\nTotal value of Updated-Changed
    ."n\n-----
-----\n\n\n";
    fwrite($fp, $report_total_up);
    // updated finish

$sum=$sum_tr+$sum_up+$sum_st;
echo " Total value of Sell is : $sum_tr <br>"
    ."Total Value of Updated products is : $sum_up <br>"
    ."Total value of Stock is : $sum_st <br><br>";

echo "<br><br><br>Total price of this branch is $sum <br><br><br>";
/*echo "The value you expected is $value";
    $c_diff=$value-$sum;
echo "<br><br>The differences between your value and branch value is
$c_diff";
*/
$report_var_total="Total value of this city Branch is = $sum\n\n ";
    fwrite($fp, $report_var_total);
fclose($fp);
?>
<br>
<br>
<a href="city_reports.rtf">download as file </a><br>
<br>

<?

```

```

    }
    else
    {
        echo "wrong operation";
        echo "<br>";
    }
?>

</div></td>
</tr>
<tr>
    <td width="45%" height="19" align="center" valign="middle">&nbsp;</td>
    <td width="34%" align="center" valign="top">&nbsp;</td>
</tr>
</table>
</body>
</html>

```



## CONCLUSION

Computerised stock control systems run on similar principles to manual ones, but are more flexible and information is easier to retrieve. You can quickly get a stock valuation or find out how well a particular item of stock is moving. A computerised system is a good option for businesses dealing with many different types of stock. Other useful features include: stock and pricing data integrating with accounting and invoicing systems. All the systems draw on the same set of data, so you only have to input the data once. Sales Order Processing and Purchase Order Processing can be integrated in the system so that stock balances and statistics are automatically updated as orders are processed; automatic stock monitoring, triggering orders when the re-order level is reached; identifying the cheapest and fastest suppliers and bar coding systems which speed up processing and recording. The software will print and read bar codes from your computer. The system will only be as good as the data put into it. Run a thorough stocktake before it goes "live" to ensure accurate figures. It's a good idea to run the previous system alongside the new one for a while, giving you a back-up. Choose a system: There are many software systems available. Talk to others in your line of business about the software they use, or contact your trade association for advice. Your needs might include: multiple prices for items, prices in different currencies, automatic updating, selecting groups of items to update, single-item updating, using more than one warehouse, ability to adapt to your changing needs, quality control and batch tracking, integration with other packages, multiple users at the same time. Avoid choosing software that's too complicated for your needs as it will be a waste of time and money.

## REFERENCES

1. <http://www.apache.com>
2. <http://www.php.net>
3. <http://www.mysql.com>
4. [www.webmaster.com](http://www.webmaster.com)
5. [www.programlama.com](http://www.programlama.com)