NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Interactive Web Page design for Car Gallery

Graduation Project
Com - 400

Student: Rami Kassim (20032584)

Supervisor: Assoc. Prof. Dr. Adil Amirjanov

Nicosia - 2007

ACKNOWLEDGMENT

All praise and glory to almighty ALLAH, the lord of the universe, who is the entire source of all knowledge and wisdom endowed to mankind. All thanks are due to him who gave me the ability and patience throughout my studies for completing this task.

I would like to acknowledge to my parents who have brought all of their efforts to support me, without knowing the return and who have patiently encouraged me to be the best everywhere.

I would like also to thank halime ozkan because she stood beside me in my project and supported me.

I would like to thanks my project supervisor Assoc. Prof. Dr. Adil Amirjanov for his intellectual support, encouragement and enthusiasm which made it possible to accomplish this project. I appreciate his most gracious encouragement and very valued constructive critics throughout my education.

My special thanks goes to NEU education staff especially to computer engineering teaching staff for their generosity and special concern of me and all COMPUTER ENGINEERING STUDENTs.

Abstract

The aim of interactive web page car gallery is to help customers to view and buy cars. The car gallery website includes many pages of html and asp. And used many techniques like HTML, ASP, JavaScript and JAVA programming language and Database, therefore user would enjoy and watch more and having some information about cars that what car gallery afford to the customer, with the help of searching, the customer can find and select the appropriate car.

TABLE OF CONTENTS

INTRODUCTION

Internet is worldwide computer network connecting millions of computers. The beauty if html is that it made the internet transparent. Clients' software called web browsers communicated with web servers.

The clients to which they are requesting are called servers. The servers contain many pages and when the client request for the particular page then it is sent.

Explanation of html, JavaScript, asp, java, database and its connection that are used in html i.e. hyper text mark up language which describes simple html elements and tags used to enclose the information in the car gallery web page that can be viewed by many users. Cascading style sheet is explained that how to make cascading style sheet and then put into my web pages so that all pages look similar. The form method is explained that how to send information to the new users, well frames is also has an important factor that can divide the pages into many columns containing each column different page or information.

Second chapter is about the JavaScript, JavaScript is a scripting language with very simple syntax that is developed by Netscape and most popular language. There are many function of JavaScript that can be used to make car gallery website more attractive and well organized. JavaScript can be put regardless of position restriction that is where we need either in body or head or in both. Similar to other programming language we can declare the variables in java and by using built in operators we can perform many complex functions. We can also declare function like other languages in JavaScript and can call at different points in the page whenever we need.

Third chapter in car gallery web page is about the Active server page the most vital role in the web page. ASP can contain same thing like html and script but here scripts run the server. The asp has many function i.e. it can dynamically edit , change, add some content to the web page and it can also retrieve the records contain in the database and put the result on the web page. For the form type request query string is used for the get method and requests. Form is used for the post method. In the asp we can also declare any variables like other programming language. We can also include the outside asp files to any page to avoid rewriting the same thing like database connection. There are many asp script and methods that make our web pages dynamic and more attractive.

Fourth chapter is about the database design, database is the most important thing while we talking about the memory. Relational database is efficient approach to the complex database that we can make the database relation in order to have less memory and fast accessing. By giving simple example, database relations are explained.

Fifth chapter is about java, java is actually more than a computer language, it's also a programming environment that includes a computer set of programming tools. To use these programming tools you need to get your own copy of the java development kit (jdk). Java can be used to create two types of programs:

- Applet
- Stand-alone application.

Sixth chapter is about the Interactive web page design for car gallery, in this chapter explanation of how database connection is established and can be dealt with the database tables using webpage, and step by step to Interactive web page design for car gallery.

CHAPTER ONE
INTRODUCTION TO HTML

This collection of pages explains how to use the different HTML document description elements, or *tags* and how to use these elements to write good, well designed HTML documents. This particular page describes the overall content and organization of the material presented here, reviews some related resources that may be of interest, and describes the meanings of the navigational "buttons" you use to navigate from page to page. HTML or Hypertext Markup Language is designed to specify the logical organization of a document, with important hypertext extensions. This choice was made because the same HTML document may be viewed by many different “browsers”, of very different abilities. Thus, for example, HTML allows you to mark selections of text as titles or paragraphs, and then leaves the interpretation of these marked elements up to the browser.
HTML instructions divide the text of a document into blocks called elements. These can be divided into two broad categories those to define how the BODY of the document is to be displayed by the browser and those to define information ‘about’ the document, such as the title or relationships to other document [1].

1.1 Using HTML

HTML consists of a series of short **codes** typed into a text-file by the site author — these are the tags. The text is then saved as a html file, and viewed through a browser, like *Internet Explorer* or Netscape Navigator. This browser reads the file and translates the text into a visible form, hopefully rendering the page as the author had intended. Writing your own html entails using tags correctly to create your vision. You can use anything from a rudimentary text-editor to a powerful graphical editor to create html pages, the tags are what separate normal text from HTML code. You might know them as the words between the <triangle-brackets>. They allow all the cool stuff like images and tables etc, just by telling your browser what to render on the page. Different tags will perform different functions. The tags themselves don't appear when you view your page through a browser, but their effects do. The simplest tags do nothing more than apply formatting to some text, like this:

<b>these words will be bold**</b>**, and these will not.

In the example above, the <b> tags were wrapped around some text, and their effect will be that the contained text will be bolded when viewed through an ordinary web browser. If you want to see a list of a load of tags to see what's ahead of you, look at this tag reference. Learning the tags themselves is dealt with in the next section of this website, My First Site.

1.2 Elements in HTML

The HTML instructions, along with the text to which the instructions apply, are called HTML elements. The HTML instructions are themselves called tags, and look like <element_name> i.e., they are simply the element name surrounded by left and right angle brackets.

Most elements mark blocks of the document for particular purpose or formatting: the <element_name>tag marks the beginning of such as section. The end of this section is the marked by the ending tag </element_name> note the leading slash character "/" that appears in front of the element name in an end tag. End, or stop tags are always indicated by this leading slash character.

1.2.1 Empty Elements

Some elements are empty that is, they do not affect a block of the document in some way. These elements do not require an ending tag. An example is the <HR>element, which draws a horizontal line across the page. This element would simply be entered as <HR>.

1.2.2 Elements with Attributes

Many elements can have arguments that pass parameters to the interpreter handling this element. These arguments are called attributes of the element. For example, consider the element A, which marks a region of text as the beginning (or end) of a hypertext link. This element can have several attributes. One of them. Href, specify this in the tag for a Write:

<A HREF=HTTP://www.neu.edu.tr>near east university</a>

Where the attribute HREF is assigned the indicated value. Note that the element is not empty, and that it is closed by the tag</a>.note else that end tags never take attributes the attributes to an element are always placed in the start tag.

1.3 HTML Document Structure

HTML document are structured into two parts, the head, and the body. Both of these are contained within the HTML element simply denotes this as an HTML document.

The head contains information about the document that is not generally displayed with the document, such as its TITLE. The BODY contains the body of the text, and is where material to be displayed. Element allowed inside the HEAD, such as TITLE, are not allowed inside the BODY, and vice [2].

Example of Document Structure

```
<HTML>
 <HEAD>
   <TITLE>Example Page</TITLE>
</HEAD>
<BODY>
  <h1>Example Project</h1>
<p> any thing </p>
<ul>
```

```
<li><A HREF="www.links.com"?>more links</a>Search Eng. </ul>
</BODY>
</HTML>
```

1.4 HTML Elements

1.4.1 Head

The HEAD contains general information, or meta-information, about the document. It is the fires thing in any document, lying above the BODY and just after the <HTML> tags starting the document.

The contents of the HEAD are not displayed as part of the document text: the displayed material is found within the BODY. Consequently, only certain mark-up elements can be placed within the HEAD. These are:

HEAD of a Document: A description of the HEAD part of an HTML document, and of the HTML `elements' valid in the HEAD.

BODY of a Document: A description of the BODY part of an HTML document - the BODY contains the part of the document actually displayed by the `browser' - and of the HTML `elements' valid in the BODY.

BASE: A record of the original URL of the document: this allows moving the document to a new directory (or even a new site) and having relative URLs access the appropriate place with respect to the original URL.

Usually placed in the HEAD by the server or a script/program to indicate that a document is searchable.

LINK: define the relationships between this document and another or other. A document can have several link elements.

TITLE: the title of the document, this element is mandatory – all documents must have a TITLE.

1.5 Additional Tags

1.5.1 Frame Tag

The <frame> tag defines what HTML document to put into each frame. The column can be set to 25% of the width of the browser window. The second column can be set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column. But it's better to but the frames pages asp extensions so that we can change easily.

1.5.2 Table Tag

Tables are defined with the <table> tag. A table is divided into rows (with the <tr>tag), and each row is divided into data cells (with the <td>tag). The letters td stands for "table data", which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, table,

Example

```
<table border="1" width="100%">
        <tr>
                <td></td>
        </tr>
</table>
```

1.5.3 Form Tag

A form is an area that can contain form elements. Form elements are elements that allow the user to enter information (like text fields, text area fields, drop-down menus, radio buttons, checkboxes, etc) in a form. A form is defined with the <form> tag.<input> the most used form tag is the <input> tag. The type of input mostly use in textfield.

Example:

```
    <form action="form_action.asp"
    method="get">

    First name:
    <input type="text" name="fname" value="Mickey" />
    <br />
    Last name:
    <input type="text" name="lname" value="Mouse" />
    <br />
    <input type="submit" value="Submit" />

    </form>
    <p>
    If you click the "Submit" button, you will send your input to a new page called
    form_action.asp.
    </p>
```

1.5.4 Style Tag

Defines a style in a document. The style element goes in the head section. If you want to include a style sheet in your page, you should define the style sheet externally, and link to it using <link>.

```
<head>
<style type="text/css">

h1 {color: red}
h3 {color: blue}
</style>
</head>
```

1.5.5 Applet Tag

DTD the attribute is allowed. S=Strict, T=Transitional, and F=Frameset. See table 1.1,

Table 1.1 descript applet tag

Attribute	**Value**	**Description**	**DTD**
height	pixels	Defines the height of the applet	TF
width	pixels	Defines the width of the object	TF

1.5.6 Script Tag

Here we should to know that Code within this element is executed immediately when the page is loaded, if it is not in a function. Script that appears after a <frameset> tag will be ignored.

```
<script type="text/javascript">
document.write("Car gallery!")
</script>
```

1.6 Simple Html Tags

As it shown in table 1.2,

Table 1.2 show some simple tags

Tag	Description
<b>	Defines bold text
<br>	Breaks the line
<tite>	Defines title of the page
<i>	Define italic text
<Small>	Defines small text
<Strong>	Defines strong text
<Sub>	Defines subscripted text
<Sup>	Defines superscripted text
<Table>	Defines table
<Frame>	Defines frame page
<Pre>	Defines preformatted text

CHAPTER TWO
JAVASCRIPT

JavaScript is it to improve the design, validate forms, and much more. JavaScript was developed by Netscape and is the most popular scripting language on the internet.
JavaScript works in all major browsers that are version 3.0 or higher. It is designed to add interactivity to HTML pages. JavaScript is a scripting language a scripting language is a lightweight programming language which contains lines of executable computer code.
A JavaScript is usually embedded directly in HTML pages and it is an interpreted language (means that scripts execute without preliminary compilation) [3].

2.1 Style Customization

Use JavaScript to determine what browser and platform you are using to view our web site. Because web browsers are not identical in the way they display data, we would like to configure fonts, colors and spacing to best fit your environment. This is done with a technology called Cascading Style Sheets. When you download a page from HP.com, the JavaScript program can choose the best style sheet for your environment, dynamically. In the absence of JavaScript, your browser will display information in its default manner, without any style sheet enhancements.

2.1.1 Image roll-overs

Many images, particularly those that are hyper-linked, will "roll over". That means that when you place your mouse pointer over the image, the image changes, and when your mouse moves off the image it returns to its original state. This is purely a visual effect to make the image look active. Without JavaScript, the hyperlinks will still work.

2.1.2 Pull-down menu navigation

Some of the pages on HP.com include pull-down menus for navigation. These consist of a drop-down input control that allows you to choose from a list of destinations. By submitting your choice, you can navigate directly to your selected destination. The pull-down menus are implemented via JavaScript and will not work unless JavaScript is enabled. However, all of the destinations in these menus are reachable via other links on our pages. The pull-down menu is merely a short-cut.

2.2 Passing information from JavaScript to HTML

A couple of ways to pass information from JavaScript to HTML, When writing HTML pages you may run into situations where you need some information from JavaScript to be passed to HTML so you can manipulate it server-side. This is a topic of discussion in many PHP forums so I have decided to present my ideas on the subject. Recently I was writing a web application using HTML and I wanted to be able to store the user's timezone in my database. This can't be done with HTML alone; I knew I would have to use JavaScript to get the timezone offset for the user and somehow pass that information to HTML so I could store it in MySQL.

I thought of a couple of ways that my problem could be solved. I could get the timezone offset in JavaScript and then use JavaScript to set a cookie that I could then read in HTML. I used this method for a while but then ran into problems with paranoid users who turned off cookies. I decided that instead of using cookies I would add a hidden form field to my login screen where I could store the timezone offset. That way, when the user logged in, they would send me their timezone offset. There are users out there who turn off their JavaScript so I made sure there was a way to manually set the timezone in the user preferences area, but for those users who leave their JavaScript turned on, this saves them a lot of time.

For those of you who wish to try the cookie method, here it is:

```
<script type="text/javascript">
var tzo=(new Date().getTimezoneOffset()/60)*(-1);
setCookie("mytzo", tzo, 30);
</script>
```

This snippet of JavaScript will get the user's timezone offset and set a cookie called "mytzo" that lasts for 30 days. The timezone offset is stored as the number of hours difference from Greenwich Mean Time (GMT). This value can then be read by PHP and stored into MySQL or another database.

The other method I spoke of is to set a hidden form element to the time zone offset value. To implement this I created a JavaScript function called getTZO() that gets called when the submit button is clicked. It gets the timezone offset and stores it in the hidden input with the id of "mytzo" so it can get submitted to my PHP page. The code is as follows:

```
function getTZO( )
{
   document.getElementById('mytzo').value =
   (new Date().getTimezoneOffset()/60)*(-1);
}
```

Now, set the onclick attribute of your submit button to be onclick="javascript:getTZO();" and you should now be able to read the "mytzo" value using your PHP script you are submitting to and save it to MySQL. I prefer this second method because some users turn off their cookies.

2.3 Using the document object

The document object is one of the most important objects of JavaScript. Let's have a look at a very simple JavaScript code. The below script writes out a line of text onto the web page:

```
document.write("Hi there. This text is written using javascript!")
```

- "document" is the object in the above example.
- "write" is the method of this object. (Think of it as the arm and legs of this object that allows it to do something-anything.

JavaScript is a language of objects, and all objects (95%) of them have both methods and properties. "Document" is just one of the many objects that essentially make up JavaScript as a language-learn these objects, and you are a JavaScript programmer! It is the object that controls the layout of a webpage-background color, text, images etc. Now, the word "write" is a method of the document object. Most objects have more than one method and property (You'll see what property is very soon), and this is true for the document object as well. Let's have a look at some of the others that the document object possesses. See table 2.1,

Table 2.1 Document object

Properties	Methods
bgColor (changes bgcolor)	write (writes something)
lastModified (gives date of document's last change)	writeln (writes in new line)
referrer (gives URL of the page that linked here)	
fgColor (changes foreground color (text))	

The columns on the left are properties of the document object. They are static attributes of the object. Let's say you want to write out the date and time of the last modification to your page onto the screen. Here's what you would do:

```
<script type="text/javascript">
var example
example=document.lastModified
document.write("This page was last modified: "+example)
</script>
```

2.4 Functions with Parameters

The beauty of functions is that it can receive data from the "outside" world and process it. The term parameter is used as a definition for "what goes into the function." You can supply different data into the function each time. What this means is that you can create one function, and use it over and over again. An example should clear up this. Lets do an example that calculates the area of a trapezoid. The formula is: (width1+width2)*height/2

```
function area(w1, w2, h){
var area=(w1+w2)*h/2
alert(area+" sq ft")
}
area(2,3,7)
area(5,7,4)
area(3,2,1)
```

- w1, w2, and h are what the function use to store the input it receives from outside the function. They are the parameters. In the first function call, w1=2, w2=3, and h=7
- You can have as many parameters as you like; of course, you have to specify that in your function. In this case, there are three: w1, w2, and h.
- Another reminder: Look at: alert(area+" sq ft"). How can we add characters? Well, in JavaScript, we could, and the result is the combination of the two strings into one.

2.5 JavaScript Position

There are some codes connecting JavaScript to html

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
```

```
</script>
</body>
</html>
```

To insert a JavaScript into an HTML page, we use the <script> tag (also use the type attribute to define the scripting language).

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>

</html>
```

The word document.write is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script type="text/javascript"> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

Note: If we had not entered the <script> tag, the browser would have treated the document.write("car site!") command as pure text, and just write the entire line on the page.

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

2.5.1 Scripts in the head section

Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

2.5.2 Scripts in the body section

Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

2.5.3 Scripts in both the body and the head section

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>

</body>
```

2.6 Web Site metrics gathering

We also use JavaScript to help us find out information about how our web site is serving customers. From this we learn what pages are popular, and where users like to go on HP.com. We also learn where they do not go or what they do not find. This information can help HP to improve our web site for our customers. Per HP.com privacy policy, this JavaScript does not record any personal or otherwise sensitive information about your visit. You may be interested in our privacy policy.

CHAPTER THREE

ACTIVE SERVER PAGERS (ASP)

An ASP file is just the same as an HTML file. An ASP file can contain text, HTML, XML, and scripts. Script in an ASP file is executed on the server and this file has the file extension “asp”.

3.1 The advantage of using ASP

Instead of CGI and Perl, those of simplicity and speed provides security since your ASP code can not be viewed from the browser, since ASP files are returned as plain HTML, they can be viewed in any browser clever ASP programming can minimize the network traffic [4].

3.2 The Basics functions of ASP

Re-usable code is a boon to developers. It's easy to manage, because it only needs to be written once; then, every time you want to do the same thing, you just call it up again. However, it does require a bit of thought and planning up front. We walk through several simple applications for re-usable code in ASP, focusing on functions.

As with every design decision, the choice to develop re-usable code is a matter of trade-off. Making the decision to develop re-usable code will help to produce more robust and maintainable solutions, but may require a little more time and thinking up front.

In many ways ASP is an unstructured language. The language itself does not constrain a developer to design something in a particular way, which again has its good points and its bad points. The emphasis is on the developer to apply hisr own style to the work. Functions provide developers with an option to produce code that can be used time and time again within a project and beyond, yet many ASP developers tend to shy away from using them extensively.

This article will introduce the basic concepts of functions within ASP, and we'll go on to produce a very useful function to demonstrate the concept. To start us off, here's a very simple piece of work: doing some very basic math to demonstrate that even with the simplest of coding tasks, functions have a real use.

It's a simple enough concept. We'll take two numbers and add them together to create a result then display this to the screen. First we'll do it the natural way. We'll code it straight out – all two lines of code!

Without functions:

```
<%

result=2+2

response.write result & "<br />"

%>
```

The Real Basics of Functions in ASP - Simplemaths

Ok, pretty simple – so let's complicate this one with by wrapping it up in a function called "simplemaths." Let's look at the function protocol first:

```
<%

function simplemaths(var1, var2)

end function

%>
```

We've created a function (which doesn't actually do anything yet). It needs two input variables (var1 and var2). We would then use the following code to call the function if both of our variables equalled 3 and 2:

```
<%

call simplemaths(3, 2)

%>
```

Simple, All that we need now is to edit our function so that it actually does something! Just as the customer requested, the function will simply add our two values together and produce the result on screen.

```
<%

'Define our function

function simplemaths(var1, var2)

 result=var1+var2

 response.write result & "<br />"

end function

'run the function

call simplemaths(3, 2)

%>
```

Note that var1 and var2 are the variable names within the function and not outside it. You could choose to use variables outside of the function too. To add 10 and 20 together using variables:

```
<%

'Define our function

function simplemaths(var1, var2)
```

```
  result=var1+var2

  response.write result & "<br />"

end function

input1=10

input2=20

call simplemaths(input1, input2)

%>
```

3.3 Connecting ASP to HTML

To connect asp server to html we have first to create an asp page and write or code to connect it to the html as shown in figure 3.1. In figure 3.1 we created asp page and then we required from html to view this asp pages [5].

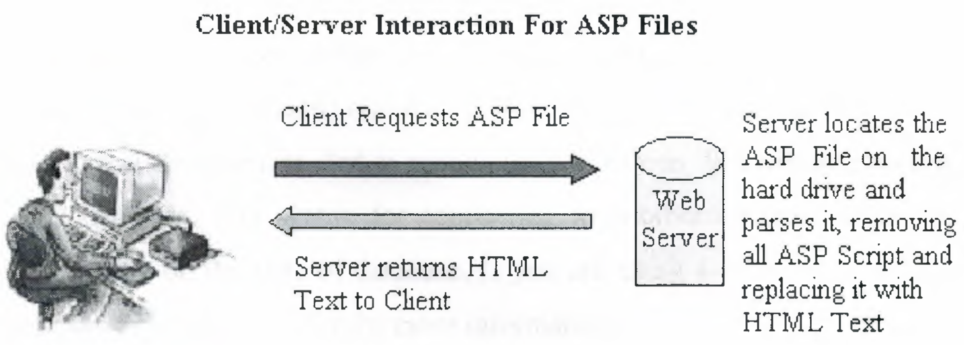

Client/Server Interaction For ASP Files
Client Requests ASP File
Server returns HTML Text to Client
Web Server
Server locates the ASP File on the hard drive and parses it, removing all ASP Script and replacing it with HTML Text

Figure 3.1

3.4 Connecting ASP pages together

In my site pages I used <form> method and hypertext method also as shown in figure 3.2.

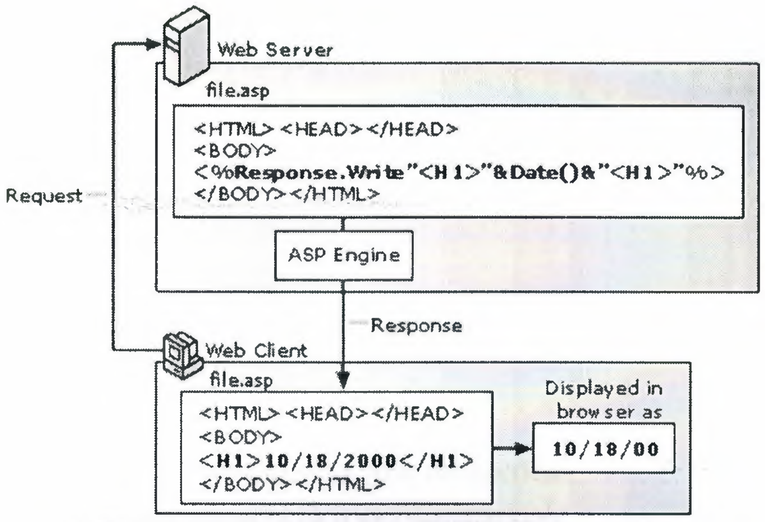


Figure 3.2 connect asp together

3.5 Connecting ASP to Database

The first option is to create a specified connection to the database. When designing my first ASP application, I set up a connection to my database in every ASP page that I needed it. Later, I found that my testing platform was different than the server, so I had to redo every one of my connections to reflect this. If I would have known more and thought about it at the time, I would have created one ASP page to hold my database connection, and then simply refer to that anytime I needed to access that connection. We will be connecting to an Access 2007 database. The syntax for connecting to another database might be a little different depending on the type of database. If you are using a different type of database, contact your systems administrator for more information.

The way that you refer to your connection is to use it as an include file. An example:

```
<%@Language=VBScript%>
<%OptionExplicit%>
<!--#includefile="connection.asp"-->
<%
```

```
setobjRS=Server.CreateObject("ADODB.Recordset")
objRS.OpenobjConn
%>
<html>....
```

The code for your connection.asp page would be as follows:

```
<%
Dim objConn 'declare your variable to hold your connection
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.ConnectionString="DRIVER={Microsoft Access Driver (*.mdb)};" &
"DBQ=c:\Inetpub\wwwroot\mydb.mdb"
objConn.open
%.
```

3.6 Writing ASP Code

When writing ASP you don't need to worry about changing your entire HTML, you simply add ASP code into your HTML pages where needed. You also don't need any special software on your computer, a simple text editor will do. To begin an ASP page you will first need to tell it what language you have written it in. The most common (and the one used in this tutorial) is VBScript. You should begin your page with:

<%@ Language=VBScript %>

All this code does is tell the ASP system that you are writing your page in VBScript. You will notice that the ASP code is enclosed in special tags. All ASP code should be enclosed in the 'percent sign tags' in the form:

<% ASP Code Here %>

Code can be written over multiple lines, but any code not enclosed in the ASP tags will simply be treated as HTML. Similarly and HTML inside these tags but not specifically sent as output by the code will cause an error.

Testing ASP

Before you start writing scripts it is a good idea to test whether ASP will run correctly on your server. Make a simple page with the following:

```
<html>
n<head><title>Test Page</title></head>
<body>
This is some HTML. Below this I have ASP<br>
<%@ Language=VBScript %><br>
Nothing should appear above here.
</body>
</html>
```

and save it as test.asp. Then upload this to your server and access it with your browser. If it has worked correctly, the page should display and you should only see the lines,
If the ASP appears in the page or the source of the page, something has gone wrong. Check the code and also the settings on your server. No ASP should appear as it should have been processed by the server before it was sent to the browser.

CHAPTER FOUR
DATABASE DESIGN

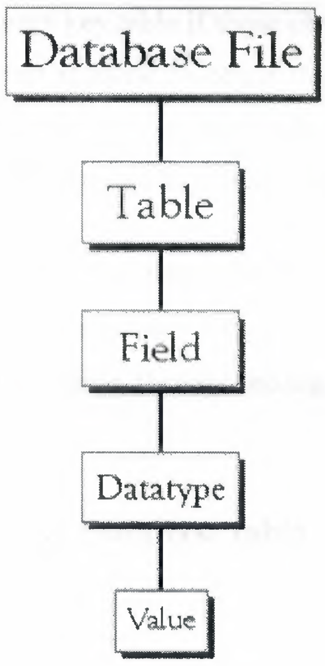
In computing, a database can be defined as a structured collection of records or data that is stored in a computer so that a program can consult it to answer queries. The records retrieved in answer to queries become information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS). The properties and design of database systems are included in the study of information science.

The term "database" originated within the computing discipline. Although its meaning has been broadened by popular use, even to include non-electronic databases, this article is about computer databases. Database-like records have been in existence since well before the industrial revolution in the form of ledgers, sales receipts and other business related collections of data.

The central concept of a database is that of a collection of records, or pieces of knowledge. Typically, for a given database, there is a structural description of the type of facts held in that database: this description is known as a schema. The schema describes the objects that are represented in the database, and the relationships among them. There are a number of different ways of organizing a schema, that is, of modelling the database structure: these are known as database models (or data models). The model in most common use today is the relational model, which in layman's terms represents all information in the form of multiple related tables each consisting of rows and columns (the true definition uses mathematical terminology). This model represents relationships by the use of values common to more than one table. Other models such as the hierarchical model and the network model use a more explicit representation of relationships [6].

4.1 Microsoft Access

- Microsoft Access is a powerful program to create and manage your databases. It has many built in features to assist you in constructing and viewing your information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works.
- First of all you need to understand how Microsoft Access breaks down a database. Some keywords involved in this process are: *Database File, Table, Record, Field, Data-type*. Here is the Hierarchy that Microsoft Access uses in breaking down a database as shown in figure 4.2.



Database File: This is your main file that encompasses the entire database and that is saved to your hard-drive or floppy disk.

Example) StudentDatabase.mdb

Table: A table is a collection of data about a specific topic. There can be multiple tables in a database.

Field: Fields are the different categories within a Table. Tables usually contain multiple fields.

Data types: Datatypes are the properties of each field. A field only has 1 datatype.

FieldName) Car Car name

Datatype) Text

Figure 4.1

4.2 Relationships in a Database

A relationship works by matching data in key fields - usually a field with the same name in both tables. In most cases, these matching fields are the primary key from one table, which provides a unique identifier for each record, and a foreign key in the other table. A foreign key (FK) is a column or combination of columns used to establish and enforce a link between the data in two tables. A link is created between two tables by adding the column or columns that hold one table's primary key values to the other table. This column becomes a foreign key in the second table.

Although the primary purpose of a foreign key is to control the data that can be stored in the foreign key table, it also controls changes to data in the primary key table. The constraint enforces referential integrity by ensuring that changes cannot be made to data in the primary key table if those changes invalidate the link to data in the foreign key table. If an attempt is made to delete the row in a primary key table or to change a primary key value, the action will fail if the deleted or changed primary key value corresponds to a value in the foreign key of another table.

To change or delete a row in a foreign key constraint successfully, you must first either delete the foreign key data in the foreign key table or change the foreign key data in the foreign key table, thereby linking the foreign key to different primary key data [7].

4.3 Create Database table

The figure 4.2 shows how to create a new database table by using Microsoft access

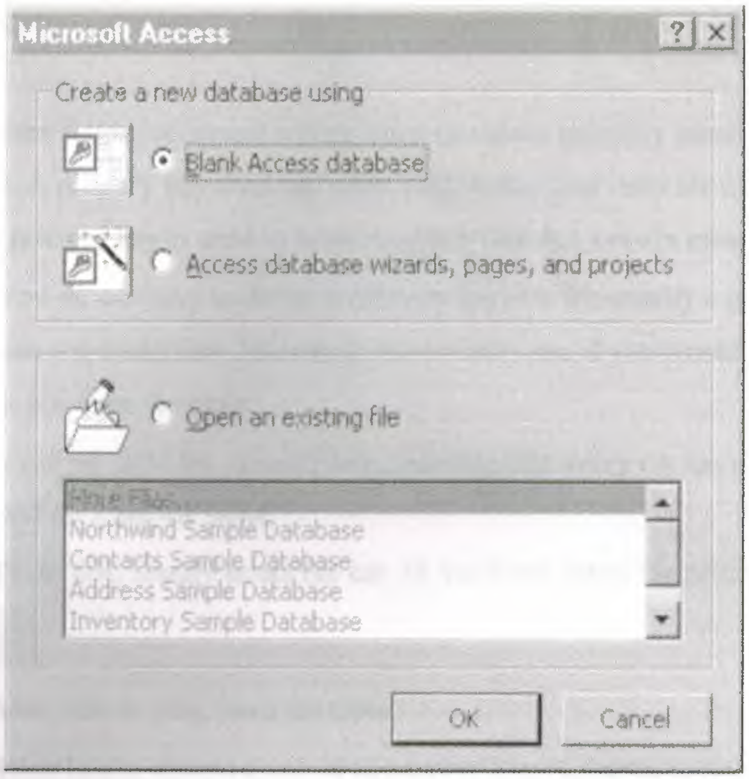

Microsoft Access
Create a new database using
[✓ Blank Access database]
Access database wizards, pages, and projects
Open an existing file
More Files...
Northwind Sample Database
Contacts Sample Database
Address Sample Database
Inventory Sample Database
OK
Cancel

Figure 4.2(a) create database

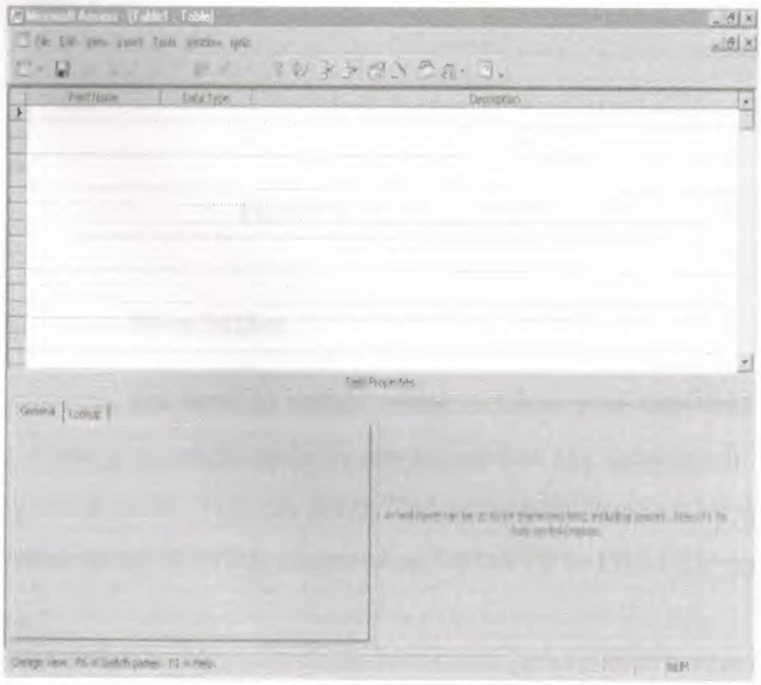
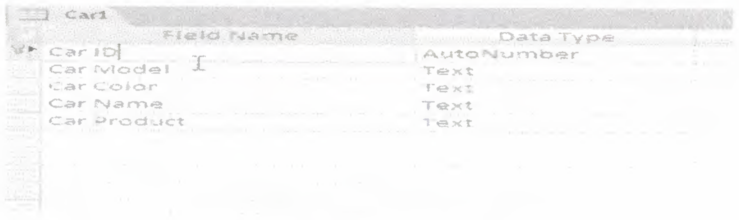


Figure 4.2(b) create database

4.3.1 Primary Key

- One or more fields (columns) whose value or values uniquely identify each record in a table. A primary key does not allow Null values and must always have a unique value. A primary key is used to relate a table to foreign keys in other tables.
- NOTE: You do not have to define a primary key, but it's usually a good idea. If you don't define a primary key, Microsoft Access asks you if you would like to create one when you save the table.
- Make the car_id field the primary key, meaning that every car has a social security number and no 2 are the same.
    - To do this, simply select the car_id field and select the primary key button
    - After you do this, Save the table

In my database table I have chosen a car_id as primary key as shown in figure 4.3.,



Car1

Field Name	Data Type
Car ID	AutoNumber
Car Model	Text
Car Color	Text
Car Name	Text
Car Product	Text

Figure 4.3 primary key

4.3.2 Update and Delete from tables

In these situations, you need to decide when to allow your application to update the database. If you allow your application to always update the database it could overwrite changes made by other users. You can control when updates succeed by specifying which columns are included in the WHERE clause of an UPDATE or DELETE statement.

```
UPDATE table ...
  SET column = newvalue
```

```
 WHERE coll = valuel
  AND col2 = value2

DELETE
 FROM table
 WHERE coll = valuel
  AND col2 = value2...
```

Controlling UPDATE's / DELETE's ,

Choose one of the following in the Where Clause for Update / Delete. The results are illustrated by an example following the table 4.1.

Table 4.1 rule of update and delete

Option	**Result**
Key Columns	The WHERE clause includes the key columns only (these are the columns you specified as the Unique or Primary Key of the table. The values in the originally retrieved key columns for the row are compared against the key columns in the database. **No other comparisons are done**. If the key values match, the update succeeds. **Caution** Be very careful when using this option, if someone else modified the same row after you retrieved it, their changes will be overwritten when you update the database. Use this option only with a single-user database or if you are using database locking. In other situations, choose one of the other two options described in this table.
Key and	The WHERE clause includes all key and updatable columns.

Updateable Columns	The values in the originally retrieved key columns and the originally retrieved updatable columns are compared against the values in the database. If any of the columns have changed in the database since the row was retrieved, the update fails.
Key and Modified Columns	The WHERE clause includes all key and modified columns. The values in the originally retrieved key columns and the modified columns are compared against the values in the database. If any of the columns have changed in the database since the row was retrieved, the update fails.

4.4 Create relationships between tables

Every table should describe only one type of objects. For instance, the Employees table contains the data about employees, their names, addresses, telephone numbers, and any other relevant data. The Customers table contains only the data that are logically related to customers. It goes without saying that the data about employees will not be kept in the Customers table. In case it is necessary to have some link between the employees and the customers, the best solution would be to create a brand new table called carcustomer. Instead of complete records, this new table will contain only unique IDs from both tables.

The same logic can be applied to the example below. In this case, the Orders table should contain data about orders. The data about the customer and about the employee who has sold the product to the customer should be entered into table 4.2.

Table 4.2 employee, customer, and order

Tables	Columns
Employees	🔑EmployeeID FirstName LastName HireDate Title
Customers	🔑CustomerID ContactName Address Country Phone
Orders	🔑OrderID 🔗CustomerID 🔗EmployeeID OrderDate

The data about every order will be entered in the Orders table. All three tables need to be connected so that the data from ID (primary key) columns from each table are kept in the Orders table, without entering all the data about customers and employees. This is achieved by setting the relationships between the tables. The Relationships Manager tool is used to accomplish this task. For the purpose of simplicity, the number of columns in each of the tables in the figure above has been reduced to minimum.

4.5 Structured Query Language

Structured Query Language (SQL) is a specialized language for updating, deleting, and requesting information from databases. SQL is an ANSI and ISO standard, and is the de facto standard database query language. A variety of established database products support SQL, including products from Oracle and Microsoft SQL Server. It is widely used in both industry and academia, often for enormous, complex databases.

In a distributed database system, a program often referred to as the database's "back end" runs constantly on a server, interpreting data files on the server as a standard relational database. Programs on client computers allow users to manipulate that data, using tables, columns, rows, and fields. To do this, client programs send SQL statements to the server. The server then processes these statements and returns replies to the client program.

Examples

To illustrate, consider a simple SQL command, SELECT. SELECT retrieves a set of data from the database according to some criteria, using the syntax:

SELECT list_of_column_names from list_of_relation_names where conditional_expression_that_identifies_specific_rows

- The list_of_relation_names may be one or more comma-separated table names, or an expression operating on whole tables
- The conditional_expression will contain assertions about the values of individual columns within individual rows within a table, and only those rows meeting the assertions will be selected. Conditional expressions within SQL are very similar to conditional expressions found in most programming languages.

4.6 SQL Data Manipulation Language (DML)

The Data Manipulation Language (DML) part of SQL permits database tables. We can also define indexes (keys), specify links between tables, and impose constraints between database tables.

The most important DML statements in SQL are:

- **SELECT** - extracts data from a database table
- **UPDATE** - updates data in a database table
- **DELETE** - deletes data from a database table
- **INSERT INTO** - inserts new data into a database table

4.6.1 SQL SELECT Statement

The SELECT statement is used to select data from a table. The tabular result is stored in a result table (called the result-set).

Syntax

```
SELECT column_name(s)
FROM table_name
```

Note: SQL statements are not case sensitive. SELECT is the same as select. The DISTINCT keyword is used to return only distinct (different) values.The SELECT statement returns information from table columns. But what if we only want to select distinct elements?

With SQL, all we need to do is to add a DISTINCT keyword to the SELECT statement:

Syntax

```
SELECT DISTINCT column_name(s)
FROM table_name
```

4.6.1.1 WHERE Clause

To conditionally select data from a table, a WHERE clause can be added to the SELECT statement.

Syntax

```
SELECT column FROM table
WHERE column operator value
```

With the WHERE clause, the following operators can be used, table 4.3,

Table 4.3 where operators

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

Note: In some versions of SQL the <> operator may be written as !=.

4.6.1.2 LIKE Clause

The LIKE condition is used to specify a search for a pattern in a column.

Syntax

```
SELECT column FROM table
WHERE column LIKE pattern
```

A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

4.6.2 INSERT INTO Statement

The INSERT INTO statement is used to insert new rows into a table.

Syntax

```
INSERT INTO table_name
VALUES (value1, value2,....)
```

You can also specify the columns for which you want to insert data:

```
INSERT INTO table_name (column1, column2,...)
   VALUES (value1, value2,....)
```

4.6.3 Update Statement

The UPDATE statement is used to modify the data in a table.

Syntax

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

```
UPDATE customer
SET Address = 'Stien 12', City = 'Stavanger'
WHERE LastName = 'Rasmussen'
```

4.6.4 DELETE Statement

The DELETE statement is used to delete rows in a table.

Syntax

```
DELETE FROM table_name
WHERE column_name = some_value
```

Delete a Row

"Nina Rasmussen" is going to be deleted:

```
DELETE FROM Person WHERE LastName = 'Rasmussen'
```

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name
or
DELETE * FROM table_name
```

Abbreviation of **s**tructured **q**uery **l**anguage, and pronounced either see-kwell or as separate letters. SQL is a standardized query language for requesting information from a database. The original version called *SEQUEL (structured **English query** language)* was designed by an IBM research center in 1974 and 1975. SQL was first introduced as a commercial database system in 1979 by Oracle Corporation.

CHAPTER FIVE

JAVA PROGRAMMING LANGUAGE

You've heard it a lot in the past several years. Everybody is saying it. But, what is all the fuss about objects and object-oriented technology? Is it real? Or is it hype? Well, the truth is--it's a little bit of both. Object-oriented technology does, in fact, exist and provide many benefits to software developers and their products. However, historically there has been a lot of hype surrounding this technology. Object-oriented concepts can be difficult to grasp and one's understanding of the concepts often slowly evolve over time. Many companies fell victim to this hardship (or took advantage of it) and claimed that their software products were object-oriented when, in fact, they weren't [8]. These false claims confused consumers causing misinformation and mistrust of object-oriented technology to abound.

However, in spite of the overuse and misuse of the term object-oriented, the computer industry is now beginning to overcome the hype and there is a growing understanding of this technology and its benefits.

What is an Object?

Objects are software bundles of data and related procedures. Software objects are often used to model real-world objects you find in everyday life.

What are Messages?

Software objects interact and communicate with each other via messages.

What are Applets?

A class is a template or prototype that defines the variables and the methods common to all objects of a certain kind.

What is Inheritance?

Classes inherit state and behaviour from their superclass. Inheritance provides a powerful and natural mechanism for organizing and structuring software programs.

Where Can I Get More Information?

This lessons given you a glimpse into the world of object-oriented design and development and may have whetted your appetite for more. Check out these fine publications to get more information about this exciting technology!

This lesson provided a basis for understanding key object-oriented terminology and concepts. Understanding these new terms and concepts is just the beginning. As you begin to design and program in the Java language, a true object-oriented language, the power of objects and classes will become apparent.

5.1 The Anatomy of an Applet

Applets are probably the most common use of the Java language today. Applets are more complicated than many Java applications because they are executed and drawn inline within Web pages, but they can access the graphics, user interface, and event structure provided by the Web browser itself. Today you learned the basics of creating applets, including the following things:

- All applets you develop using Java inherit from the Applet class, which is part of the java.applet package. The Applet class provides basic behavior for how the applet will be integrated with and reacts to the browser and various forms of input from that browser and the person running it. By subclassing Applet, you have access to all that behavior.
- Applets have five main methods, which are used for the basic activities an applet performs during its life cycle: init (), start(), stop(), destroy(), and paint(). Although you don't need to override all these methods, these are the most common methods you'll see repeated in many of the applets you'll create in this book and in other sample programs.
- To run a compiled applet class file, you include it in an HTML Web page by using the <APPLET> tag. When a Java-capable browser comes across <APPLET>, it loads and runs the applet described in that tag. Note that to publish Java applets on the World Wide Web alongside HTML files you do not need special server software; any plain old Web server will do just fine.

- Unlike applications, applets do not have a command line on which to pass arguments, so those arguments must be passed into the applet through the HTML file that contains it. You indicate parameters in an HTML file by using the <PARAM> tag inside the opening and closing <APPLET> tags. <PARAM> has two attributes: NAME for the name of the parameter, and VALUE for its value. Inside the body of your applet (usually in init()), you can then gain access to those parameters using the getParameter() method.

This lesson discusses the parts of an applet. If you haven't yet compiled an applet and included it in an HTML page, you might want to do so now. Step by step instructions are in Getting Started: The "Hello World" Applet.

Every applet is implemented by creating a subclass of the Applet class. The following figure shows the inheritance hierarchy of the Applet class. This hierarchy determines much of what an applet can do and how, as you'll see on the next few pages.

```
java.lang.Object
  |
  +----java.awt.Component
         |
         +----java.awt.Container
                |
                +----java.awt.Panel
                       |
                       +----java.applet.Applet
```

A Simple Applet

Below is the source code for an applet called Simple. The Simple applet displays a descriptive string whenever it encounters a major milestone in its life, such as when the user first visits the page the applet's on. The pages that follow use the Simple applet and build upon it to illustrate concepts that are common to many applets. If you find yourself baffled by Java source code, you might want to go to Writing Java Programs to learn about the language.

```
import java.awt.Graphics;

public class Simple extends java.applet.Applet {

   StringBuffer buffer = new StringBuffer();
```

5.1.1 Initialization

Initialization occurs when the applet is first loaded (or reloaded), similarly to the main() method in applications. The initialization of an applet might include reading and parsing any parameters to the applet, creating any helper objects it needs, setting up an initial state, or loading images or fonts. To provide behavior for the initialization of your applet, override the init() method in your applet class:

```
public void init() {
   ...
}

   public void init() {
        resize(500, 20);
      addItem("initializing... ");
   }
```

5.1.2 Starting

After an applet is initialized, it is started. Starting is different from initialization because it can happen many different times during an applet's lifetime, whereas initialization happens only once. Starting can also occur if the applet was previously stopped. For example, an applet is stopped if the reader follows a link to a different page, and it is started again when the reader returns to this page. To provide startup behavior for your applet, override the `start()` method:

```
public void start() {
    ...
}
```

Functionality that you put in the `start()` method might include creating and starting up a thread to control the applet, sending the appropriate messages to helper objects, or in some way telling the applet to begin running. You'll learn more about starting applets on Day 10, "Simple Animation and Threads."

```
  public void start() {
     addItem("starting... ");
  }
```

5.1.3 Stopping

Stopping and starting go hand in hand. Stopping occurs when the reader leaves the page that contains a currently running applet, or you can stop the applet yourself by calling stop(). By default, when the reader leaves a page, any threads the applet had started will continue running. By overriding stop(), you can suspend execution of these threads and then restart them if the applet is viewed again:

```
public void stop() {
   ...
}
```

```
  public void stop() {
     addItem("stopping... ");
```

```
}
```

5.1.4 Destroying

Destroying enables the applet to clean up after itself just before it is freed or the browser exits-for example, to stop and remove any running threads, close any open network connections, or release any other running objects. Generally, you won't want to override destroy() unless you have specific resources that need to be released-for example, threads that the applet has created. To provide clean-up behavior for your applet, override the destroy() method:

```
public void destroy() {
  ...
}
```

Technical Note
How is destroy() different from finalize(), which was described on ,"More About Methods"? First, destroy() applies only to applets. finalize() is a more general-purpose way for a single object of any type to clean up after itself.

```
public void destroy() {
      addItem("preparing for unloading...");
}

public void addItem(String newWord) {
      System.out.println(newWord);
      buffer.append(newWord);
      repaint();
}
```

5.1.5 Painting

Painting is how an applet actually draws something on the screen, be it text, a line, a colored background, or an image. Painting can occur many thousands of times during an applet's life cycle (for example, after the applet is initialized, if the browser is placed behind another window on the screen and then brought forward again, if the browser window is moved to a different position on the screen, or perhaps repeatedly, in the case of animation). You override the paint() method if your applet needs to have an actual appearance on the screen (that is, most of the time). The paint() method looks like this:

```
public void paint(Graphics g) {
    ...
}
```

Note that unlike the other major methods in this section, paint() takes an argument, an instance of the class Graphics. This object is created and passed to paint by the browser, so you don't have to worry about it. However, you will have to make sure that the Graphics class (part of the java.awt package) gets imported into your applet code, usually through an import statement at the top of your Java file:

```
import java.awt.Graphics;

   public void paint(Graphics g) {
        g.drawRect(0, 0, size().width - 1, size().height - 1);
        g.drawString(buffer.toString(), 5, 15);
   }
}
```

For comparison, here's the Alpha 3 version of Simple.java.

5.2 The Life Cycle of an Applet

You can use the Simple applet to learn about the milestones in every applet's life.

5.3 Creating Applets

For the most part, all the Java programs you've created up to this point have been Java applications-simple programs with a single `main()` method that create objects, set instance variables, and run methods. Today and in the next few days you'll be creating applets exclusively, so you will need a good grasp of how an applet works, the sorts of features an applet has, and where to start when you first create your own applets.

New Term
Java's Abstract Windowing Toolkit (awt) provides classes and behavior for creating GUI-based applications in Java. Applets make use of many of the capabilities in the awt.

5.4 Methods for Adding UI Components

Applets inherit from the AWT Container class. This means that they are designed to hold Components -- user interface objects such as buttons, labels, pop-up lists, and scrollbars. Like other Containers, applets use layout managers that control the positioning of Components.

5.5 Threads in Applets

Even the simplest applets run in multiple threads, although it's not always apparent. Many applets create and use their own threads, so that they perform well without affecting the performance of the applet viewer or of other applets.

What Applets Can and Can't Do

Things applets can't do, for security reasons: They can't load libraries written in any language but Java. Also, windows that an applet brings up are distinguished by a red bar and some warning text, so that applets can't look like trusted applications.

Things that applets can do, that you might not expect: Although most applets stop running once you leave their page, they don't have to.

5.6 Creating Applet Steps by Step

This lesson walks you through the steps necessary to integrate native code with Java programs.

The example used throughout this lesson implements the canonical "Hello World!" program. The "Hello World!" program has two Java classes: the first implements the main() method for the overall program, and the second, called HelloWorld, has one method, a native method, that displays "Hello World!". The implementation for the native method is provided in the C programming language.

Step 1: Write the Java Code

Create a Java applet, named animate, which declares a method. Also, write the main applet that creates a animate object method.

Step 2: Compile the Java Code

Use javac to compile the Java code that you wrote in Step 1.

Step 3: Create the animate.class File

Step 4: Write the C Function

Write the implementation for the native method in a C source file. The implementation will be a regular C function that's integrated with your Java class.

Step 5: Create a Dynamically Loadable Library

Use the C compiler to compile the .h file, the stubs file, and the .c file that you created in Steps 3, 4, and 5 into a dynamically loadable library.

Step 6: Run the Program

And finally,

5.7 Inserting an Applet on a Web Page

After you create a class or classes that contain your applet and compile them into class files as you would any other Java program, you have to create a Web page that will hold that applet by using the HTML language. There is a special HTML tag for including applets in Web pages; Java-enabled browsers use the information contained in that tag to locate the compiled class files and execute the applet itself. In this section, you'll learn about how to put Java applets in a Web page and how to serve those files to the Web at large.

Note
The following section assumes that you have at least a passing understanding of writing HTML pages. If you need help in this area, you may find the book Teach Yourself Web Publishing with HTML in 14 Days useful. It is also from Sams.net and also by Laura Lemay, the author of much of this book.

A simple HTML page

```
<HTML>
<HEAD>
<TITLE>This page has an applet on it</TITLE>
</HEAD>
<BODY>

<BR><APPLET CODE="Animate.class" WIDTH=200 HEIGHT=50>

</APPLET>
```

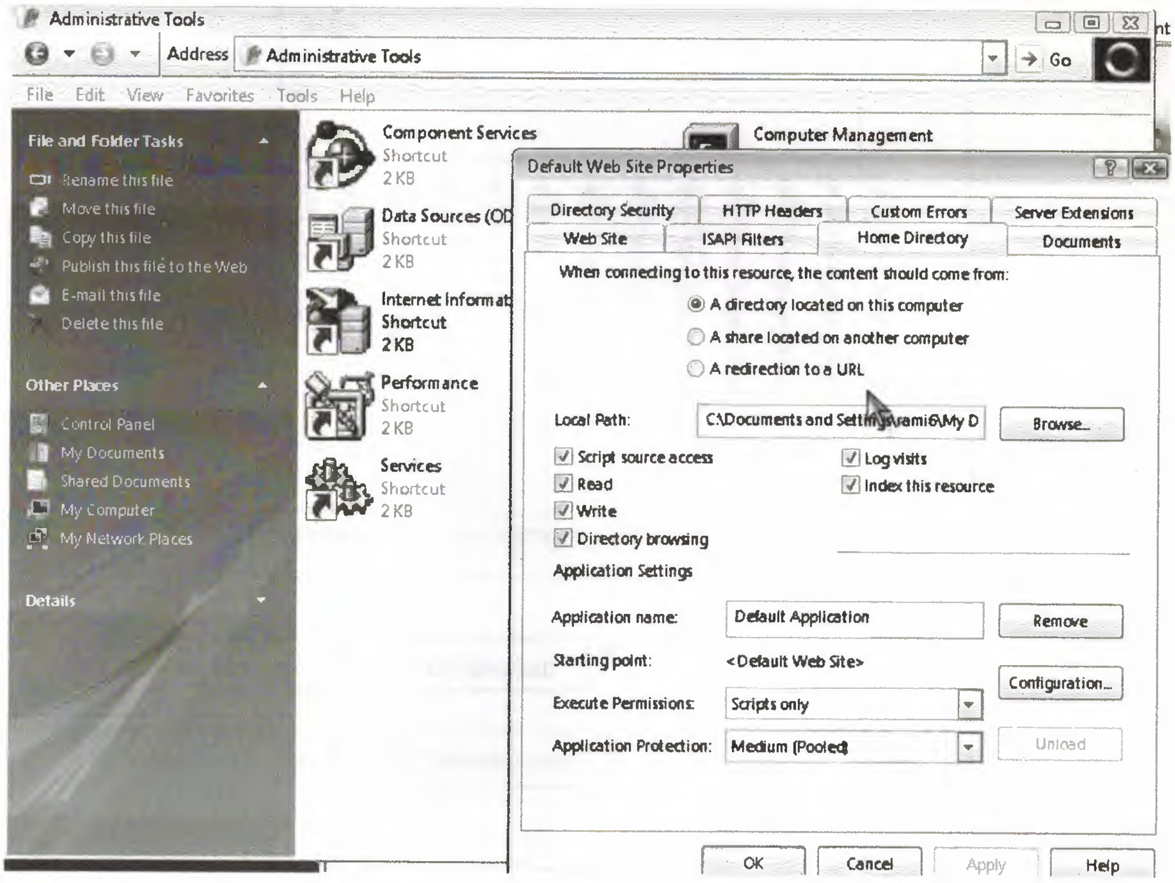
```
</BODY>
</HTML>
```

Use html to run the program [9].

CHAPTER SIX

INTERACTIVE WEB PAGE DESIGN FOR CAR GALLERY

In order to run project we first have to set or ISS server's web directory to the folder where or website pages contain as shown in figure 6.1,



Default Web Site Properties

When connecting to this resource, the content should come from:

[✓ A directory located on this computer]
A share located on another computer
A redirection to a URL

Local Path: C:\Documents and Settings\rami6\My D

[✓] Script source access
[✓] Read
[✓] Write
[✓] Directory browsing
[✓] Log visits
[✓] Index this resource

Application Settings

Application name: Default Application

Starting point: <Default Web Site>

Execute Permissions: Scripts only

Application Protection: Medium (Pooled)

Figure 6.1 set asp

6.1 Flow diagrams for Car Site

This diagram we will see how the pages connect each other and how it it work together for online as shown in figure 6.2

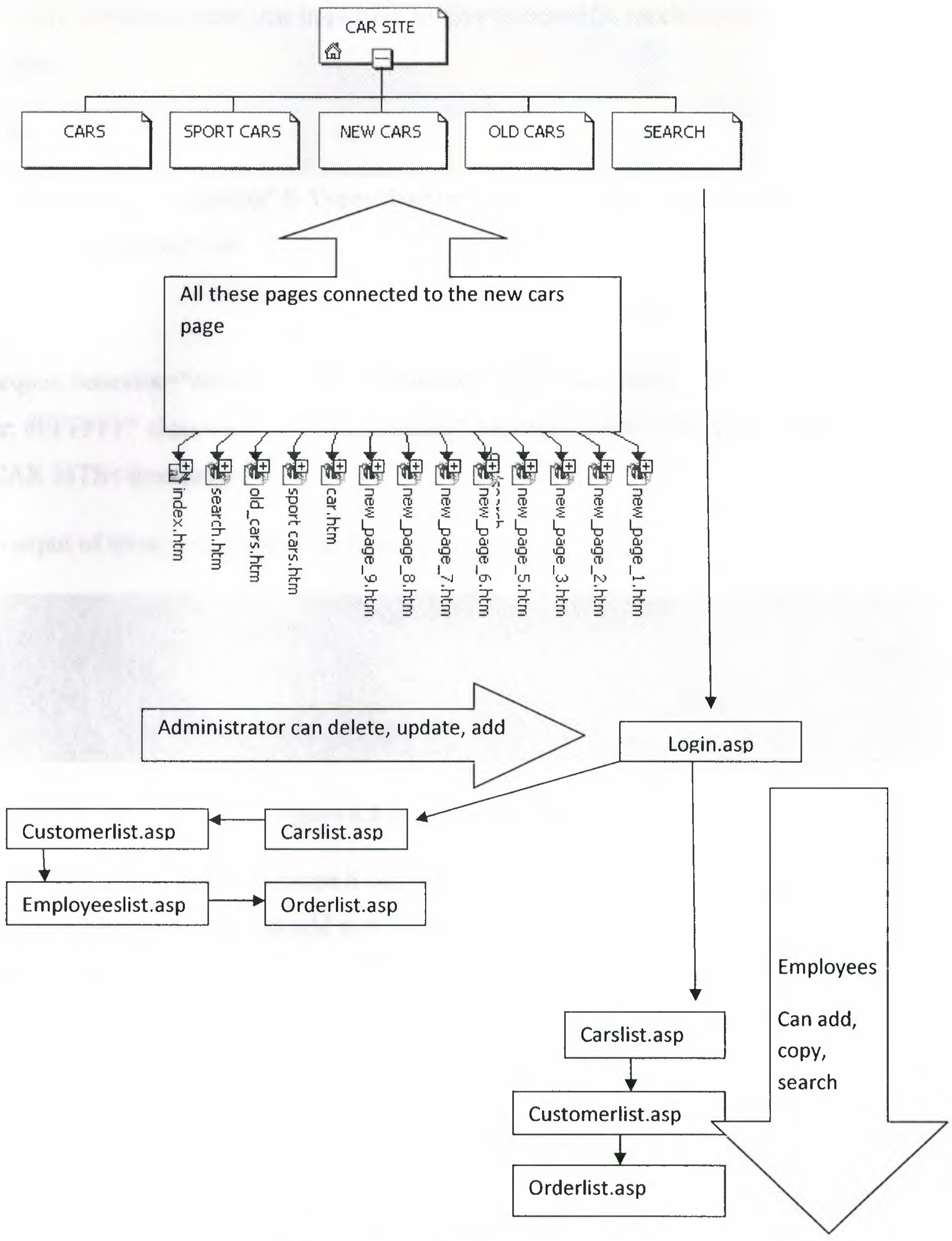


Figure 6.2: Flow Diagram

6.2 HTML & Index.html

There are some codes that important to give to beautiful touch on the html for examples:

Car site

```
<!--webbot bot="Navigation" S-Type="banner" S-Orientation="horizontal" S-Rendering="graphics" -->
```

And

```
<marquee behavior="alternate" style="font-size: 12pt; font-family: Comic Sans MS; color: #FFFFFF" class="ms-MWSInstantiated" bgcolor="#800000">WELCOME IN CAR SITE</marquee>
```

The output of these codes shown in figure 6.3,

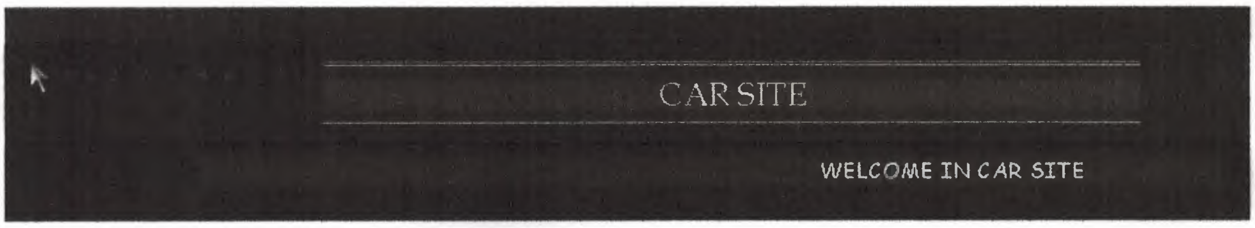


Figure 6.3 shows JavaScript

The HTML wide language because it connect many things at the same time and gives some nice touch and can add many things also like ASP and java and JavaScript [9].

```
<script language="JavaScript">
   <!--
function click() {
if (event.button==2) {
```

```
alert('Rami Kassim');
}
}
document.onmousedown=click
// -->
</script>
<style>
.spanstyle {
        position:absolute;
        visibility:visible;
        top:-50px;
        font-size:10pt;
        font-family:comic sans ms;
    font-weight:bold;
        color:red;
}
</style>
<script>
var x,y
var step=20
var flag=0
// Your snappy message. Important: the space at the end of the sentence!!!
var message="Welcome!!!"
message=message.split("")

var xpos=new Array()
for (i=0;i<=message.length-1;i++) {
        xpos[i]=-50
}
var ypos=new Array()
for (i=0;i<=message.length-1;i++) {
        ypos[i]=-50
}
function handlerMM(e){
        x = (document.layers) ? e.pageX : document.body.scrollLeft+event.clientX
        y = (document.layers) ? e.pageY : document.body.scrollTop+event.clientY
        flag=1
}
function makesnake() {
        if (flag==1 && document.all) {
```

```
for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step
            ypos[i]=ypos[i-1]
}
      xpos[0]=x+step
      ypos[0]=y
      for (i=0; i<message.length-1; i++) {
      var thisspan = eval("span"+(i)+".style")
      thisspan.posLeft=xpos[i]
            thisspan.posTop=ypos[i]
}
}
else if (flag==1 && document.layers) {
for (i=message.length-1; i>=1; i--) {
            xpos[i]=xpos[i-1]+step
            ypos[i]=ypos[i-1]
}
      xpos[0]=x+step
      ypos[0]=y
      for (i=0; i<message.length-1; i++) {
```

```
                var thisspan = eval("document.span"+i)
                thisspan.left=xpos[i]
                        thisspan.top=ypos[i]
        }
        }
                var timer=setTimeout("makesnake()",30)
}
</script>
<script>
<!-- Beginning of JavaScript -
for (i=0;i<=message.length-1;i++) {
   document.write("<span id='span"+i+"' class='spanstyle'>")
        document.write(message[i])
   document.write("</span>")
}
if (document.layers){
        document.captureEvents(Event.MOUSEMOVE);
}
document.onmousemove = handlerMM;
// - End of JavaScript - -->
```

```
</script>
```

6.3 Java in Car Site

In car site I have used java to give so nice movement on the page like[8].

```
import java.awt.*;
public class Animate extends javax. swing.JApplet implements Runnable{
Image [] picture = new Image[6];
int totalPictures = 0;
int current = 0;
Thread runner;
int pause = 500;
public void init() {
 for (int i=0;i<6; i ++){
 String imageText = null;
imageText = getParameter("image"+i);
if (imageText !=null){
totalPictures ++;
picture[i] = getImage(getCodeBase(), imageText);
} else
  break;
}
String pauseText = null;
pauseText = getParameter("pause");
if (pauseText != null){
pause = Integer.parseInt(pauseText);
  }
}
public void paint(Graphics screen){
Graphics2D screen2D = (Graphics2D) screen;
if (picture [current] != null)
 screen2D.drawImage(picture [current],0,0,this);
}
public void start() {
if (runner == null){
runner = new Thread(this);
runner.start();
 }
}
public void run(){
Thread thisThread = Thread.currentThread();
while (runner == thisThread){
repaint();
current ++;
if (current>= totalPictures)
current = 0;
try {
Thread.sleep(pause);
} catch (InterruptedException e){}
 }
  }
public void stop() {
if (runner != null){
runner = null;
  }
 }
}
```

The above code work as getting the six pictures and display them by looping one after the other as thread method very useful to display the picture clearly.

6.4 Database in Car Site

Here figure 6.3 shows the relations between tables as table customer has relation with table order and table order details has relation with order details as table order details has also relation with product all these tables show calculation to output that show information about product and time and product that have taken by customer [10].

```
SELECT
 Customers.Payment,
 Customers.Fax,
 [Order Details].ProductID,
 Orders.CustomerID,
 Customers.[Customer Name],
 Category.CategoryName,
 Products.ProductName,
 Cars.CarName,
 Employees.EmployeeID,
 Orders.OrderDate
FROM
 (((Cars
 INNER JOIN Customers ON (Cars.CarName = Customers.CarName))
 INNER JOIN Orders ON (Customers.CustomerID = Orders.CustomerID))
 INNER JOIN Employees ON (Orders.EmployeeID = Employees.EmployeeID))
 INNER JOIN [Order Details] ON (Orders.OrderID = [Order Details].OrderID),
 Products
 INNER JOIN Category ON (Products.CategoryID = Category.CategoryID),
 Models,
 Trademarks
```

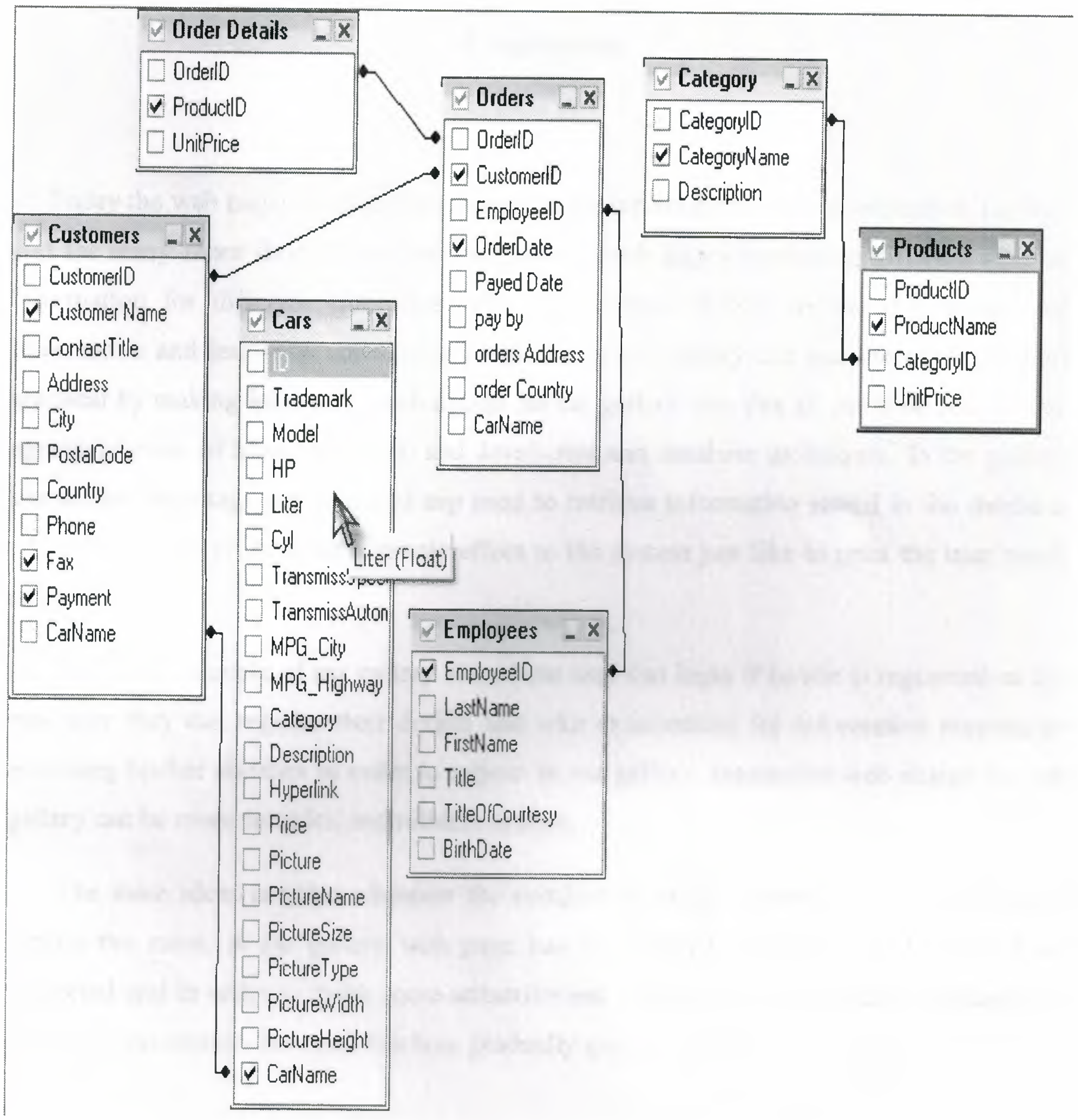


Figure 6.4: The relationship between the tables.

Conclusion

Today the web pages are the vital source for the information, trading, education, tourism and for many more fields. There are millions of web pages containing different kind of information for different categories. The idea behind is how to use that source for informative and less time consuming purposes. In car gallery site put the same thing in practical by making interactive web design for car gallery site, this all could be possible by appropriate use of html, asp, java, and JavaScript and database techniques. In car gallery site simple html tags are used and asp used to retrieve information stored in the database while JavaScript gives more dynamic effect to the system just like to print the user result etc.

This is the example of car gallery site where user can login if he/she is registered or for new user they can register their details and take examination for informative purpose or accessing his/her abilities in order to appear in car gallery ,Interactive web design for car gallery can be more complex and reliable system.

The main ideas are that whatever the complexity of the system the basic techniques remain the same. A car gallery web page has their own importance which can not be neglected and in order to make more attractive and effective web pages the techniques of html, asp, JavaScript, java and database gradually getting complex.

REFERENCFES

[1] Lan Graham, Introduction to html, Fourth Edition, 2000.

[2] Deitel, Internet & how to program, 2005.

[3] www.javascript.com

[4] Eric A. smith, Active Server Pages, 2002.

[5] www.asp.com

[6] www.databases.about.com

[7] Ibrahim Mouhsen, Learn Microsoft Access, 1990.

[8] Assist. Prof. Dr. Adil Amirjanov, java programming, Peason Eduction, 2005.

[9] www.arabjavascript.com

[10] www.aspmaker.com