

**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**INTERACTIVE WEB PAGE DESIGN FOR CAR  
GALLERY USING ASP.NET AND C#**

**Graduation Project  
COM- 400**

**Student: Hüseyin Rauf DİLSİZ (20002434)**

**Supervisor: Assist. Professor Dr Adil AMIRJANOV**

**Lefkoşa – 2006**

## ACKNOWLEDGEMENT

First I want to thank Mr. Adil AMİRJANOV to be my advisor. During my education, he helped me so much in order to handle many difficulties successfully. I learned a lot about programming languages from him. Whenever I ask him questions, he explains my questions friendly, patiently. With his assistance, I developed myself very quickly.

Again I should thank Mr Rahib ABIYEV to his patient and kinds that shows us in our education life he spent to much efforts to make us a good engineer. I want present him to my endless respects.

I also want thank to my friends, Engin, Mesut, Göktuğ, Sinan, Ozan, Süleyman and the others who help me in preparation of my project. With their patient and kind help, I could cope with some problems which I encountered while preparing my project. Thanks to Faculty of Engineering.

Finally, I want to thank my family, especially my fiancée Yıldız and my sister Simla. Apart from their endless patient and support, I could not manage to finish my education.

## ABSTRACT

Today many people use internet to do their works such as bank operation, searching for source, communication and buying or selling something. As we know the E-Commerce is the most important side of World Wide Web and the easiest way to reach the customers whose are interested in your products. Nowadays most of the companies try to reach their customers on the web as like as second hand vehicle utilization galleries.

People can easily search for something on the internet. They can also easily compare two or more things on the internet. So it is available and easy way for them to shopping on the internet

The project is a web application program which depends on buying or selling second hand vehicle. The vehicle details are presented on the web site which visitors or customer determined. Visitors or customers can easily search the cars by price, model or both of them. The registered users, who have the account, can also sell their vehicle on the web site. The another thing on the site is, there is a special entry for administrators, they can delete user accounts or vehicles, they can change "contact us", "about us", and "link" pages data and the last thing is they reach and erase the messages which are sent by visitors or members.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENT</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>TABLE OF CONTENTS</b>	<b>iii</b>
<b>INTRODUCTION</b>	<b>1</b>
<b>CHAPTER -1-HTML</b>	<b>2</b>
<b>1.1 Basic Tags</b>	<b>4</b>
<b>1.2 Character Style</b>	<b>5</b>
<b>1.3 Working with HTML Controls</b>	<b>6</b>
<b>1.4 Types of Styles and Style Sheets</b>	<b>8</b>
<b>1.5 Style Properties</b>	<b>8</b>
<b>CHAPTER -2-ASP.NET</b>	<b>10</b>
<b>2.1 .NET</b>	<b>12</b>
<b>2.2 Introduction to Web Forms</b>	<b>13</b>
<b>2.3 Web Controls</b>	<b>14</b>
<b>2.4 Basic Web Controls</b>	<b>15</b>
<i>2.4.1 Label</i>	<b>15</b>
<i>2.4.2 TextBox</i>	<b>16</b>
<i>2.4.3 Button</i>	<b>17</b>
<i>2.4.4 Image</i>	<b>17</b>

2.4.5 <i>ImageButton</i>	18
2.4.6 <i>LinkButton</i>	18
2.4.7 <i>HyperLink</i>	18
2.4.8 <i>RadioButton</i>	19
2.4.9 <i>RadioButtonList</i>	19
2.4.10 <i>CheckBox</i>	20
2.4.11 <i>CheckBoxList</i>	20
2.4.12 <i>DropDownList</i>	21
2.4.13 <i>ListBox</i>	21
2.4.14 <i>Panel</i>	22
2.4.15 <i>PlaceHolder</i>	23
<b>CHAPTER -3-C# LANGUAGE</b>	24
3.1 <b>Object-Oriented Programming</b>	25
3.2 <b>C# Language Fundamentals</b>	27
3.3 <b>Types</b>	28
3.4 <b>C# and the .NET Framework</b>	30
3.5 <b>The .NET Platform</b>	30
3.6 <b>The .NET Framework</b>	31
3.7 <b>Classes, Objects, And Types</b>	32
3.8 <b>Namespaces</b>	33
<b>CHAPTER -4-Microsoft SQL Server 2000</b>	34
4.1.1 <i>Enterprise Manager</i>	34

4.1.2 <i>Query Analyzer</i>	34
4.1.3 <i>SQL Profiler</i>	34
4.1.4 <i>Service Manager</i>	34
4.1.5 <i>Data Transformation Services (DTS)</i>	34
<b>4.2 SQL Server Architecture</b>	35
<b>4.3 Fundamentals of SQL Server 2000 Architecture</b>	36
<b>4.4 Relational Database Components</b>	36
4.4.1 <i>Database</i>	37
4.4.2 <i>Relational Database</i>	37
4.4.3 <i>Scalable</i>	38
<b>4.5 Structured Query Language</b>	39
<b>4.6 Database Applications and Servers</b>	39
<b>4.7 Server Database Systems</b>	40
<b>4.8 Advantages of Server Database System</b>	40
<b>4.9 Advantages of SQL Server 2000 as a Database Server</b>	42
<b>4.10 Desktop Database Systems</b>	43
<b>CHAPTER -5-OVERVIEW TO CMC CAR GALLERY PROJECT</b>	45
<b>5.1 Web Page Structure of User Side</b>	45
5.1.1 <i>Top Side of The Pages</i>	46
5.1.2 <i>Home Page</i>	46
5.1.3 <i>Search Pages</i>	47
5.1.4 <i>Sign In Page</i>	48



5.1.5 Forget Password Page	49
5.1.6 Create New User Page	50
5.1.7 User Operations Page	51
5.1.8 Sell a Vehicle	52
5.1.9 Send Message page	53
5.1.10 Update or Delete User Page	54
5.1.11 Update or Delete Vehicle Page	55
5.1.12 Site Map	56
5.1.13 Links Page	57
5.1.14 About Us Page	58
5.1.15 Contact Us Page	58
5.1.16 Result Page	59
<b>5.2 Web Page Structure of Administrator Side:</b>	<b>59</b>
5.2.1 Administrator Sign In Page	60
5.2.2 Administrator Operation Page	61
<b>5.3 Creating Database &amp; Database Schema</b>	<b>62</b>
5.3.1 Tables	63
5.3.2 Table Relations:	77
5.3.3 Database Environment:	78
5.3.3.1 Database Access:	78
5.3.4 SQL Query Analyzer	79

<b>5.4 Web Page Design (Code Side)</b>	<b>80</b>
<i>5.4.1 HTML Code Analyzing</i>	<b>80</b>
<i>5.4.2 ASP.NET Code Analyzing</i>	<b>82</b>
<i>5.4.3 C# Code Analyzing</i>	<b>88</b>
<b>CONCLUSION</b>	<b>93</b>
<b>REFERENCES</b>	<b>94</b>



## INTRODUCTION

E-Commerce is very important in web because, most of the companies or people want to earn more money on the web with their internet sites.

The main aim of this project is to prove that, how the most efficient languages can be used for E-commerce and how to form (design) a web page for second hand vehicle utilize gallery on the web with using ASP.NET and C# in code behind and MS SQL Server 2000.

Chapter -1- includes fundamentals of HTML (Hyper Text Markup Language), its tags, types, character styles and how to work with HTML controls in ASP.NET. Also you will get some properties of HTML styles.

Chapter -2- gives a small brief about .NET technology and history of ASP.NET also its working procedure, web forms which depend on ASP.NET and web controls. In this chapter it gives some basic codes with C# definitions.

Chapter -3- answers the question of “what is C#” and also “What is Object-Oriented Programming”. This chapter will show you the fundamentals of C#, its types, C# and .NET frameworks, classes, object and type and also namespaces.

Chapter -4- is information about the Microsoft SQL Server 2000 database, which is used in the project. You can get the explanation of SQL server architecture, its fundamentals, database applications and server and also its advantages.

Chapter -5- is overview of the CMC Car Gallery web page. In this chapter is also analyzing the web design project and shows how to build a web page for second hand vehicle utilization galleries with HTML, ASP.NET, C# and MICROSOFT SQL 2000 SERVER.

Finally, conclusion section presents the important results obtained within project.

## CHAPTER -1-

### HTML

HTML (Hyper Text Markup Language) is a web programming language designed to create web documents or web pages. Based upon SGML (Standard Generalized Markup Language), HTML's basic concept involves the use of "tags". These "tags", "mark up" or alert the browser that the document contains hypertext so it can be interpreted and rendered as a web page document. All HTML documents consist of a mix and match of HTML "tags" and "regular" text. Tags only aid in describing the document content or text, and thus leave the actual appearance and layout decisions for a web browser to handle when the web page is rendered or opened.

HTML was originally developed by Tim Berners-Lee while at CERN, and popularized by the Mosaic browser developed at NCSA. During the course of the 1990s it has blossomed with the explosive growth of the Web. During this time, HTML has been extended in a number of ways. The Web depends on Web page authors and vendors sharing the same conventions for HTML. This has motivated joint work on specifications for HTML.

HTML 2.0 (November 1995, see [RFC1866]) was developed under the aegis of the Internet Engineering Task Force (IETF) to codify common practice in late 1994. HTML+ (1993) and HTML 3.0 (1995, see [HTML30]) proposed much richer versions of HTML. Despite never receiving consensus in standards discussions, these drafts led to the adoption of a range of new features. The efforts of the World Wide Web Consortium's HTML Working Group to codify common practice in 1996 resulted in HTML 3.2 (January 1997, see [HTML32]).

Most people agree that HTML documents should work well across different browsers and platforms. Achieving interoperability lowers costs to content providers since they must develop only one version of a document. If the effort is not made, there is much greater risk that the Web will devolve into a proprietary world of incompatible formats, ultimately reducing the Web's commercial potential for all participants.

Each version of HTML has attempted to reflect greater consensus among industry players so that the investment made by content providers will not be wasted and that their documents will not become unreadable in a short period of time.

HTML has been developed with the vision that all manner of devices should be able to use information on the Web: PCs with graphics displays of varying resolution and color depths, cellular telephones, hand held devices, devices for speech for output and input, computers with high or low bandwidth, and so on.

HTML documents are plain-text files that can be created using any basic or high-level text editor, such as Notepad, TextPad, or Microsoft Word or any other HTML authoring program. When you create an HTML document, you must save it with a *.html* or *.htm* extension. By default, most text editors save documents with a *.txt* extension, which is not capable of being displayed by a web browser. The *.html* or *.htm* extension allows the document to be rendered and displayed by a browser.

Unfortunately for web designers, web documents are browser dependent; sometimes, different browsers display content differently. A document may look crisp and clean in Internet Explorer, but it may have a slightly different look in Netscape, or vice versa. Web page designers should make every attempt to create portable HTML documents that can be opened by many different web browsers while showing little or no visual differences. Reliable HTML web pages are created by following all syntax rules and understanding which tags are supported by all web browsers.

It is very easy to create a web page, a simple text editor, and a web browser. It's that simple. In most cases, a web page, or web site which is a collection of related web pages, should be designed locally on a computer first, and then once completed, the web documents and files may be uploaded for publishing on the World Wide Web. This makes web site or web page creation, extremely easier than trying to edit existing documents on the WWW. To learning actual HTML code, programmers should become familiar with the following short listing of terms:

- HTML - short for HyperText Markup Language; basic programming language of the World Wide Web based upon SGML (Standard Generalized Markup Language).



- Web browser - application capable of interpreting and rendering HTML code and other web programming languages.
- URL - [Uniform Resource Locator] - the address to any web site or web page document that is part of the World Wide Web.
- Hyperlink, link - text, image or object in a web page document that "links" or "points" to another document on the World Wide Web.
- Element - a fundamental component of structure in a web document; web pages *are ultimately divided into many individual elements*.
- Tag - used to denote various elements in a document; it signals a command or instruction for a web browser and specifically describes the type of content.
- Attribute - additional information included inside the start tag of an element; issues a command to a web browser telling what kind of operation is required.
- Web document - actual web page text file with an extension of *.html* or *.htm* that is capable of being displayed by a browser.

## 1.1 Basic Tags

Tags are elements of the HTML language. Almost every kind of tag has an opening symbol and a closing symbol. For example, the <HEAD> tag identifies the beginning of heading information. It also has a closing tag </HEAD>.

### **<HTML></HTML> tag**

This element tells browsers that the file is a HTML document. Each HTML document starts with the tag <HTML>. This tag should be first thing in the document. It has an associate closing tag </HTML> which must be the last tag in the file.

### **<HEAD></HEAD> tag**

The head contains important information about the document.

## **<TITLE></TITLE> tag**

The title tag is an important tag. It is used to display a title on the top of your browser window. Both the opening and the closing tags go between the head tags.

## **<BODY></BODY> tag**

The Body Tag is used to identify the start of the main portion of your webpage. Between <BODY> </BODY> tags you will place all images, links, text, paragraphs, and forms. We will explain each tag that is used within the body of the HTML file.

### **1.2 Character Style**

Character styles include physical and logical character styles, and Face, Size, and Color. The following is character style table 1.1.

Table 1.1

Type	Choice	function
Physical styles	<B>	Make text <b>bold</b> .
	<I>	Make text <i>italic</i> .
	<U>	Make text <u>underline</u> .
	<Strike>	Make text <del>strikethrough</del> .
	<Sup>	Make text <sup>superscript</sup> .
	<Sub>	Make text <sub>subscript</sub> .
	Teletype	Make text teletype.
Logical styles	<Strong>	Indicate the text is very important.
	<Em>	Indicate the text is important.
	<Cite>	Indicate that the text is from a book or other document.
	<Address>	Indicate that the text is an address.
	<Dfn>	Indicate that the text is a definition.
	<Samp>	Indicate that the text is a sequence of literal characters.

	Keyboard	Indicate that the text is keyboard input.
	<Var>	Indicate that the text is a variable.
	<Code>	Indicate that the text is code.
Face	Default	Make text display in the default font (Times New Roman) of the Web browser.
	Family	Type a list of fonts separated by commas (for example, Helvetica, Arial, Courier). The text will display in the first listed font found on the browser's system.
	(Font name)	Make the text display in the font specified. (If the font is not available on the browser's system, another font will be substituted.)
Size	1 through 7 (3 is the default)	Format text with 7 sizes where 7 is the largest size and 1 is the smallest.
	Increase	Format text with the largest size (same as 7).
	Decrease	Format text with the smallest size (same as 1).
Color	"#xxxxxx" or: White, Red, Blue and Others	Make the text a different color.

### 1.3 Working with HTML Controls

HTML controls are outwardly identical to plain old HTML 4.0 tags, but employ the `runat="server"` attribute. For each of HTML's most common tags, a corresponding server-side HTML control exists, although Microsoft has added a few tags and some extra properties for each. Creating HTML controls is easy—we simply stick a `runat="server"` attribute on the end of a normal HTML tag to create the HTML control version of that tag. The complete list of current HTML control classes and their associated tags is given in the next table 1.2.

HTML control classes are all contained within the `System.Web.UI.HtmlControls` namespace.

Because HTML controls are processed on the server side by the ASP.NET runtime, we can easily access their properties through code elsewhere in the page. If you're familiar with JavaScript, HTML, and CSS, then you'll know that manipulating



text within HTML tags, or even manipulating inline styles within an HTML tag, can be cumbersome and error-prone. HTML controls aim to solve this by allowing you to manipulate the page easily with your choice of .NET language, for instance, using VB.NET or C#. We'll start by looking at the HTML controls library on the next table.

Table 1.2

Class	Associated Tags Class
HtmlAnchor	<a href="..." runat="server">
HtmlButton	<button runat="server">
HtmlForm	<form runat="server">
HtmlImage	<img runat="server">
HtmlInputButton	<input type="submit" runat="server"> <input type="reset" runat="server"> <input type="button" runat="server">
HtmlInputCheckBox	<input type="checkbox" runat="server">
HtmlInputFile	<input type="file" runat="server">
HtmlInputHidden	<input type="hidden" runat="server">
HtmlInputImage	<input type="image" runat="server">
HtmlInputRadioButton	<input type="radio" runat="server">
HtmlInputText	<input type="text" runat="server">
HtmlSelect	<select runat="server">
HtmlTable	<table runat="server">
HtmlTableRow	<tr runat="server">
HtmlTableCell	<td runat="server"> <th runat="server">
HtmlTextArea	<textarea runat="server">
HtmlGenericControl	All other HTML tags, including <span runat="server"> <div runat="server"> <body runat="server"> <font runat="server">

## 1.4 Types of Styles and Style Sheets

There are three different ways of associating styles to elements of a particular Web page. It has been already mentioned the first, and usually the best, which is an external file:

**External File:** By placing your **style rules** in an external style sheet, you can link this one file to any Web pages where you want those styles to be used. This makes updating a Website's overall look a cakewalk.

**Document Wide:** Rather than having an external sheet, you can place style rules for a page within a `<style>` tag inside that page's head element. The problem is that we can't then use those styles in another page without typing them in again, which makes global changes to the entire site difficult to manage.

**Inline:** Inline styles allow us to set styles for a single tag using the style attribute. For instance, we might create a text box in regular HTML with a style attribute that draws a border around the text box.

**Classes:** Arguably the most popular way to use styles within your pages, classes allows you to set up a custom style that will be applied to any tag or control that has a class attribute that matches the name of your custom style.

**Tag Redefinition:** Redefining a tag affects the appearance of certain standard HTML tags. For instance, the `<hr>` tag is generally given a width of 100% by default, but you could redefine the tag in CSS to have a width of 50%.

## 1.5 Style Properties

There are many different types of properties that you can modify using style sheets. Below is a list of the common types:

**Font:** This category provides you with the ability to format text level elements, including their font face, size, decoration, weight, color, etc.

**Background:** This category allows you to customize backgrounds for objects and text. Modifying these values gives you control over the color, image, and whether or not you want to repeat an image.

**Block:** This category allows you to modify the spacing between paragraphs, lines of text, and spaces between text and words.

**Box:** The box category provides changes and customizations for tables. If you need to modify borders, padding, spacing, and colors on a table, row, or cell, you can modify elements within this category.

**Border:** This category lets you draw boxes of different colors, styles and thicknesses around page elements.

**List:** This category allows you to customize the way ordered and unordered lists are created.

**Positioning:** Modifying positioning allows you to move and position tags and controls freely.

## CHAPTER -2-

### ASP.NET

For years now, Active Server Pages (ASP) has been arguably the leading choice for Web developers building dynamic Websites on Windows Web servers. ASP has gained popularity by offering the simplicity of flexible scripting via several languages. That, combined with the fact that it's built into every Microsoft Windows-based Web server, has made ASP a difficult act to follow.

Early in 2002, Microsoft released its new technology for Internet development. Originally called ASP+, it was finally released as ASP.NET, and represents a leap forward from ASP both in sophistication and productivity for the developer. It continues to offer flexibility in terms of the languages it supports, but instead of a range of simple scripting languages, developers can now choose between several fully-fledged programming languages. Development in ASP.NET requires not only an understanding of HTML and Web design, but also a firm grasp of the concepts of object-oriented programming and development.

ASP.NET is a server-side technology for developing Web applications based on the Microsoft .NET Framework.

ASP.NET is server-side; that is, it runs on the Web server. Most Web designers start by learning client-side technologies like HTML, JavaScript, and Cascading Style Sheets (CSS). When a Web browser requests a Web page created with client-side technologies, the Web server simply grabs the files that the browser (the client) requests and sends them down the line. The client is entirely responsible for reading the code in the files and interpreting it to display the page on the screen. Server-side technologies, like ASP.NET, are different. Instead of being interpreted by the client, server-side code (for example, the code in an ASP.NET page) is interpreted by the Web server. In the case of ASP.NET, the code in the page is read by the server and used dynamically to generate standard HTML/JavaScript/CSS that is then sent to the browser. As all processing of ASP.NET code occurs on the server, it's called a server-side technology. As Figure shows, the user (client) only sees the HTML, JavaScript, and CSS within the browser.



The server (and server-side technology) is entirely responsible for processing the dynamic portions of the page.

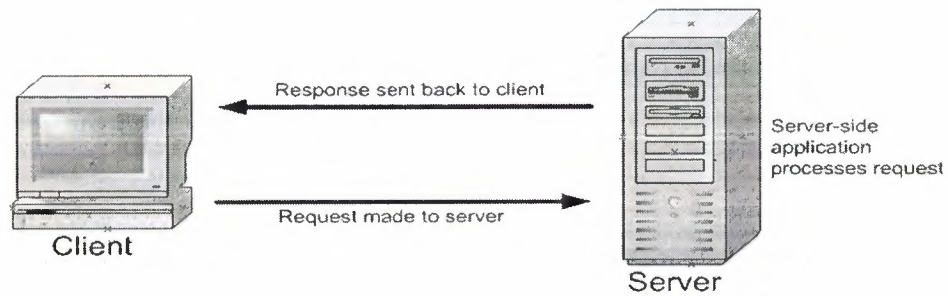


Figure The Web server is responsible for processing the server-side code and presenting the output to the user (client).

ASP.NET is a technology for developing Web applications. A Web application is just a fancy name for a dynamic Website. Web applications usually store information in a database on the Web server, and allow visitors to the site to access and change that information. Many different programming technologies and supported languages have been developed to create Web applications; PHP, JSP (using Java), CGI (using Perl), and ColdFusion (using CFML) are just a few of the more popular ones. Rather than tying you to a specific technology and language, however, ASP.NET lets you write Web applications using a variety of familiar programming languages.

Finally, ASP.NET is based on the Microsoft .NET Framework. The .NET Framework collects all the technologies needed for building Windows applications, Web applications, and Web Services into a single package with a set of more than twenty programming languages. To develop Websites with ASP.NET, you'll need to download

the .NET Framework Software Development Kit. If we compare with other options for building Web applications, ASP.NET has the following advantages:

ASP.NET lets you use your favorite programming language, or at least one that's really close to it. The .NET Framework currently supports over twenty languages, four of which may be used to build ASP.NET Websites.

ASP.NET pages are compiled not interpreted. Instead of reading and interpreting your code every time a dynamic page is requested, ASP.NET compiles dynamic pages into efficient binary files that the server can execute very quickly. This represents a big

in performance when compared with the technology's interpreted predecessor, ASP.

ASP.NET has full access to the functionality of the .NET Framework. Support for XML, Web Services, database interaction, email, regular expressions, and many other technologies are built right into .NET, which saves you from having to reinvent the wheel.

ASP.NET allows you to separate the server-side code in your pages from the HTML layout. When you're working with a team composed of programmers and design specialists, this separation is a great help, as it lets programmers modify the server-side code without stepping on the designers' carefully crafted HTML and vice versa.

With all these advantages, ASP.NET has relatively few downsides. In fact, only two come to mind:

ASP.NET is a Microsoft technology. While this isn't a problem in itself, it does mean that, at least for now, you need to use a Windows server to run an ASP.NET Website. Serious ASP.NET development requires an understanding of object-oriented programming.

## 2.1 .NET

.NET is the result of a complete make-over of Microsoft's software development products, and forms part of the company's new strategy for delivering software as a service. The key features that .NET offers include:

**.NET Platform:** The .NET platform includes the .NET Framework and tools to build and operate services, clients, and so on. ASP.NET, is a part of the .NET Framework.

**.NET Products:** .NET products currently include MSN.NET, Office.NET, Visual Studio.NET, and Windows Server 2003, originally known as Windows .NET Server. This suite of extensively revised systems provides developers with a friendly, usable environment in which they may create applications with a range of programming languages including C++. NET, Visual Basic. NET, ASP.NET, and C#. Because all these



products are built on top of .NET, they all share key components, and underneath their basic syntaxes you'll find they have much in common.

**.NET My Services:** An initiative formerly known as "Hailstorm", .NET My Services is a set of XML Web Services currently being provided by a host of partners, developers, and organizations that are hoping to build corporate services and applications for devices and applications, as well as the Internet.

The collection of My Services currently extends to passport, messenger, contacts, email, calendars, profiles, lists, wallets, location, document stores, application settings, favorite Websites, devices owned, and preferences for receiving alerts.

## **2.2 Introduction to Web Forms**

With the inception of new technologies, there's always new terminology to master. ASP.NET is no different. With ASP.NET, even the simplest terms that were previously used to describe a Web page have changed to reflect the processes that occur within them. Before we begin to describe the process followed by Web Forms, now discuss the foundation concept of Web pages.

On the most basic level, a Web page is a text file that contains markup. Web pages are meant to be viewed from a browser window, which parses the file containing markup to present the information to the user in the layout envisaged by the developer. Web pages can include text, video, sound, animations, graphics, and even chunks of "code" from a variety of technologies.

An HTML form, is a page that contains one or more form elements grouped together within an HTML <form> tag. Users interact with the various form elements to make certain choices, or provide certain information; this information is then sent to the server for processing upon the click of a submit button. This is useful to us as ASP.NET developers because regular HTML forms have a built-in mechanism that allows forms to be submitted to the server. Once the form has been submitted, some kind of extra technology in this case, ASP.NET needs to be present on the server to perform the actual form processing.

In ASP.NET, we call Web pages Web Forms; they contain presentational elements (ASP.NET Web controls) in an HTML form, as well as any code (the processing logic) we've added for the page's dynamic features.

## 2.3 Web Controls

Web Forms allow users to interact with our site using **Web controls**. With Web controls, Microsoft basically reinvented HTML from scratch. For example, it created two different Web controls that correspond to the two different versions of the HTML `<select>` tag: a `DropDownList` control and a `ListBox` control. This means there isn't a direct one-to-one correspondence between the Web controls and standard HTML tags, as there is with HTML controls. Web controls follow the same basic pattern as HTML tags, but the tag name is preceded by `asp:` and the name is capitalized using "CamelCasing." Consider the HTML `<input>` tag, which creates an input text box on screen:

```
<input type="text" name="username" size="30" />
```

The equivalent Web control is the `TextBox` control, and it would look like this:

```
<asp:TextBox id="username" Columns="30" runat="server">
</asp:TextBox>
```

Note that, unlike many HTML tags, Web controls always require a closing tag (the `</asp:TextBox>` part above). We can also use the shorthand `/>` syntax if our Web control tag doesn't contain anything between its opening and closing tags. So, we could also write this `TextBox` like so:

```
<asp:TextBox id="username" Columns="30" runat="server" />
```

To sum up, the key points to remember when working with Web controls are:

- \*All Web controls must be placed within a `<form runat="server">` tag to function properly.

- \*All Web controls require `id` and `runat="server"` properties to function properly.

- \*All Web controls follow the same pattern, but different properties (attributes) are available to different controls.

- \*They all start with the `asp` prefix, followed by a colon.

There are more Web controls than HTML controls, and some offer advanced features that simply aren't available in HTML alone. Which are:

- \*Basic Web controls

- \*Validation Web controls

- \*Data controls

- \*User controls

- \*Rich controls

## 2.4 Basic Web Controls

The basic Web controls perform the on-screen layout of a Web page, and mirror in many ways the HTML controls that are based on regular HTML. However, they offer some new refinements and enhancements, and should be used in place of HTML whenever possible.

### 2.4.1 Label

The easiest way to display static text on your page is simply to add the text to the body of the page without enclosing it in any tag. However, if you want to modify the text displayed on a page from ASP.NET code, you can display your text within a Label control. Here's a typical example:

```
<asp:Label id="lblMessage" Text="" runat="server" />
```

The following code sets the Text property of the Label control to display the text “Hello World”:

```
C#  
public void Page_Load() {  
    lblMessage.Text = "Hello World";  
}
```

Reading this Page\_Load() handler code, we can see that when the page first loads, the Text property of the Label control with the ID of lblMessage will be set to “Hello World.”

### 2.4.2 TextBox

The TextBox control is used to create on screen a box in which the user can type or read standard text. This Web control can be set to display a standard *HTML text input field*, an *HTML password field*, or an *HTML text area*, using the TextMode property. The following code shows how we might use it in a simple login page:

```
<p>Username:  
<asp:TextBox id="txtUser" TextMode="SingleLine" Columns="30"  
runat="server" /></p>  
<p>Password:  
<asp:TextBox id="txtPassword" TextMode="Password" Columns="30"  
runat="server" /></p>  
<p>Comments:  
<asp:TextBox id="txtComments" TextMode="MultiLine" Columns="30"  
Rows="10" runat="server" /></p>
```

In each of the three instances above, the attribute TextMode dictates the kind of text box to render.



### 2.4.3 Button

By default, the Button control renders the same form submit button that's rendered by the HTML `<input type="Submit">` tag. When a button is clicked, the form containing the button is submitted to the server for processing, and both click and command events are raised. The following code displays a Button control and a Label:

```
<asp:Button id="btnSubmit" Text="Submit" runat="server"
OnClick="WriteText" />
<asp:Label id="lblMessage" runat="server" />
```

Notice the OnClick attribute on the control. Unlike the HtmlButton HTML control, OnClick assigns a *server-side* event handler—there is no need to remember to use OnServerClick. When the button is clicked, the Click event is raised and the WriteText() subroutine is called. The WriteText() subroutine will contain the code that performs the intended function for this button, such as displaying a message for the user:

```
C#
public void WriteText(Object s, EventArgs e) {
    lblMessage.Text = "Hello World";
}
```

It's important to realize that most Web controls have events associated with them, and the basic idea and techniques are the same as for the Click event of the Button control.

### 2.4.4 Image

An Image control places on the page an image that can be accessed dynamically from code; it equates to the `<img>` tag in HTML. Here's an example:

```
<asp:Image id="myImage" ImageUrl="mygif.gif" runat="server"
AlternateText="description" />
```

### 2.4.5 ImageButton

An ImageButton control is similar to a Button control, but it uses an image you supply in place of the typical gray Windows-style button. For example:

```
<asp:ImageButton id="myImgButton" ImageUrl="myButton.gif"
runat="server" />
```

### 2.4.6 LinkButton

A LinkButton control renders a hyperlink on your page. From the point of view of ASP.NET code, LinkButtons can be treated in much the same way as buttons, hence the

```
<asp:LinkButton id="myLinkButon" Text="Click Here" runat="server"/>
```

### 2.4.7 HyperLink

The HyperLink control, which is similar to the LinkButton control, creates a hyperlink on your page. It's simpler and faster to process than LinkButton, but, unlike the LinkButton control, which offers features such as Click events and validation, HyperLink can be used only to click and navigate from one page to the next.

```
<asp:HyperLink id="myLink" NavigateUrl="http://www.example.com/"
ImageUrl="myButton.gif" runat="server">My Link</asp:HyperLink>
```

The ImageUrl attribute, if specified, causes the control to display a linked image instead of the text provided.



## 2.4.3 RadioButton

You can add individual radio buttons to your page one by one, using the `RadioButton` control. Radio buttons are grouped together using the `GroupName` property. Only one `RadioButton` control from each group can be selected at a time.

```
<asp:RadioButton id="radSanDiego" GroupName="City"
Text="San Diego" runat="server" />
<asp:RadioButton id="radBoston" GroupName="City" Text="Boston"
runat="server" />
<asp:RadioButton id="radPhoenix" GroupName="City" Text="Phoenix"
runat="server" />
<asp:RadioButton id="radSeattle" GroupName="City" Text="Seattle"
runat="Server" />
```

The main event associated with `RadioButtons` is the `CheckChanged` event; which can be handled with the `OnCheckChanged` attribute.

## 2.4.4 RadioButtonList

Like the `RadioButton` control, the `RadioButtonList` control represents radio buttons. However, the `RadioButtonList` control represents a list of radio buttons and uses more compact syntax. Here's an example:

```
<asp:RadioButtonList id="radFavColor" runat="server">
<asp:ListItem Text="Red" Value="red" />
<asp:ListItem Text="Blue" Value="blue" />
<asp:ListItem Text="Green" Value="green" />
</asp:RadioButtonList>
```

One of the great features of the RadioButtonList is its ability to bind to a data source. For instance, imagine you have a list of employees in a database. You could create a page that binds a selection from that database to the RadioButtonList control, to list dynamically certain employees within the control. The user would then be able to select one (and only one) employee from that list, and our code could determine the choice. The most useful event produced by RadioButtonList is the SelectedIndexChanged event, to which you can assign a handler with the OnSelectedIndexChanged attribute

#### **2.4.10 CheckBox**

You can use a CheckBox control to represent a choice that can be only a yes (checked) or no (unchecked) value.

```
<asp:CheckBox id="chkQuestion" Text="I like .NET!" runat="server"/>
```

As with the RadioButton control, the main event associated with a CheckBox is the CheckChanged event; which can be handled with the OnCheckChanged attribute.

#### **2.4.11 CheckBoxList**

The CheckBoxList control represents a group of check boxes; it's equivalent to using several CheckBox controls in row:

```
<asp:CheckBoxList id="chk1FavDrinks" runat="server">
  <asp:ListItem Text="Pizza" Value="pizza" />
  <asp:ListItem Text="Tacos" Value="tacos" />
  <asp:ListItem Text="Pasta" Value="pasta" />
</asp:CheckBoxList>
```

Like the RadioButtonList control, the CheckBoxList control has the capability to bind to a data source, and produces a SelectedIndexChanged event that you can handle with OnSelectedIndexChanged.

## 2042 DropDownList

A DropDownList control is similar to the HTML <select> tag. The DropDownList control allows you to select one item from a list using a drop-down menu.

```
<asp:DropDownList id="ddlFavColor" runat="server">
  <asp:ListItem Text="Red" value="red" />
  <asp:ListItem Text="Blue" value="blue" />
  <asp:ListItem Text="Green" value="green" />
</asp:DropDownList>
```

As is the case with other collection-based controls, such as the CheckBoxList and RadioButtonList controls, the DropDownList control can be bound to a database, thus allowing you to extract dynamic content into a drop-down menu. The main event produced by this control, as you might expect, is SelectedIndexChanged, handled with UnselectedIndexChanged.

## 2043 ListBox

A ListBox control equates to the HTML <select> tag with the size attribute set to one or more. The ListBox control allows you to select items from a multiline menu. If you set the SelectionMode attribute to Multiple, the user will be able to select more than one item from the list, as in this example:

```
<asp:ListBox id="listTechnologies" runat="server"
  SelectionMode="Multiple">
  <asp:ListItem Text="ASP.NET" Value="aspnet" />
  <asp:ListItem Text="JSP" Value="jsp" />
  <asp:ListItem Text="PHP" Value="php" />
  <asp:ListItem Text="C#" Value="csharp" />
  <asp:ListItem Text="Coldfusion" Value="cf" />
```

```
</asp:ListBox>
```

Again, because the ListBox control is a collection-based control, it can be dynamically bound to a data source. The most useful event that this control provides is, you guessed it, `SelectedIndexChanged`, with the corresponding `OnSelectedIndexChanged` attribute.

#### 2.4.14 Panel

The Panel control functions similarly to the `<div>` tag in HTML, in that the set of items that resides within the tag can be manipulated as a group. For instance, the Panel could be made visible or hidden by a Button's Click event:

```
<asp:Panel id="pnlMyPanel" runat="server">
  <p>Username:
  <asp:TextBox id="txtUsername" Columns="30" runat="server" />
</p>
  <p>Password:
  <asp:TextBox id="txtPassword" TextMode="Password"
  Columns="30" runat="server" /></p>
</asp:Panel>
<asp:Button id="btnHide" Text="Hide Panel" OnClick="HidePanel"
runat="server" />
```

The code above creates two TextBox controls within a Panel control. The Button control is outside of the panel. The `HidePanel()` subroutine would then control the Panel's visibility by setting its `Visible` property to `False`:

```
C#
public void HidePanel(Object s, EventArgs e) {
    pnlMyPanel.Visible = false;
```



```
}
```

In this case, when the user clicks the button, the Click event is raised and the HidePanel() subroutine is called, which sets the Visible property of the Panel control to False.

#### **2.4.15 Placeholder**

The Placeholder control lets us add elements at a particular place on a page at any time, dynamically, through code.

```
<asp:Placeholder id="phMyPlaceholder" runat="server" />
```

The following code dynamically adds a new HtmlButton control within the placeholder.

```
C#  
public void Page_Load() {  
    HtmlButton btnButton = new HtmlButton();  
    btnButton.InnerText = "My New Button";  
    phMyPlaceholder.Controls.Add(btnButton);  
}
```

## CHAPTER -3-

### C# LANGUAGE

The C# language is disarmingly simple, with only about 80 keywords and a dozen built-in data types, but C# is highly expressive when it comes to implementing modern programming concepts. C# includes all the support for structured, component-based, object-oriented programming that one expects of a modern language built on the shoulders of C++ and Java.

At the heart of any object-oriented language is its support for defining and working with classes. Classes define new types, allowing you to extend the language to better model the problem you are trying to solve. C# contains keywords for declaring new classes and their methods and properties, and for implementing encapsulation, inheritance, and polymorphism, the three pillars of object-oriented programming.

In C# everything pertaining to a class declaration is found in the declaration itself. C# class definitions do not require separate header files or Interface Definition Language (IDL) files. Moreover, C# supports a new XML style of inline documentation that greatly simplifies the creation of online and print reference documentation for an application.

C# also supports interfaces, a means of making a contract with a class for services that the interface stipulates. In C#, a class can inherit from only a single parent, but a class can implement multiple interfaces. When it implements an interface, a C# class in effect promises to provide the functionality the interface specifies.

C# also provides support for structs, a concept whose meaning has changed significantly from C++. In C#, a struct is a restricted, lightweight type that, when instantiated, makes fewer demands on the operating system and on memory than a conventional class does. A struct can't inherit from a class or be inherited from, but a struct can implement an interface.

C# provides component-oriented features, such as properties, events, and declarative constructs (called attributes). Component-oriented programming is supported by the CLR's support for storing metadata with the code for the class. The metadata describes the class, including its methods and properties, as well as its security needs and



other attributes, such as whether it can be serialized; the code contains the logic necessary to carry out its functions. A compiled class is thus a self-contained unit; therefore, a hosting environment that knows how to read a class' metadata and code needs no other information to make use of it. Using C# and the CLR, it is possible to add custom metadata to a class by creating custom attributes. Likewise, it is possible to read class metadata using CLR types that support reflection.

An assembly is a collection of files that appear to the programmer to be a single dynamic link library (DLL) or executable (EXE). In .NET, an assembly is the basic unit of reuse, versioning, security, and deployment. The CLR provides a number of classes for manipulating assemblies.

A final note about C# is that it also provides support for directly accessing memory using C++ style pointers and keywords for bracketing such operations as unsafe, and for warning the CLR garbage collector not to collect objects referenced by pointers until they are released.

**Before starting C# features there will be small brief of Object-Oriented Programming because of C# language is also an OOP Language.**

### ***3.1 Object-Oriented Programming***

The concept of objects and instances in computing had its first major breakthrough with the PDP-1 system at MIT which was probably the earliest example of capability based architecture. Another early example was Sketchpad made by Ivan Sutherland in 1963; however, this was an application and not a programming paradigm.

Objects as programming entities were introduced in Simula 67, a programming language designed for making simulations, created by Ole-Johan Dahl and Kristen Nygaard of the Norwegian Computing Centre in Oslo. (Reportedly, the story is that they were working on ship simulations, and were confounded by the combinatorial explosion of how the different attributes from different ships could affect one another. The idea occurred to group the different types of ships into different classes of objects, each class of objects being responsible for defining its own data and behavior.) Such an approach was a simple extrapolation of concepts earlier used in analog programming. On analog computers, such direct mapping from real-world phenomena/objects to analog

phenomena/objects (and conversely), was (and is) called 'simulation.' Simula not only introduced the notion of classes, but also of instances of classes, which is probably the first explicit use of those notions.

The Smalltalk language, which was developed at Xerox PARC, introduced the term Object-oriented programming to represent the pervasive use of objects and messages as the basis for computation. Smalltalk creators were influenced by the ideas introduced in Simula 67, but Smalltalk was designed to be a fully dynamic system in which objects could be created, modified, and 'consumed' "on the fly" rather than having a system based on static objects. It also introduced the notion of 'inheritance.' (Thus, Smalltalk was clearly a major move beyond the analog programming models, which made no use of "instances of classes," or even Simula, which made no use of the "inheritance property.")

The ideas in Simula 67 were also used in many other languages, from derivatives of Lisp to Pascal.

Object-oriented programming developed as the dominant programming methodology during the mid-1980s, largely due to the influence of C++, an extension of the C programming language. Its dominance was further cemented by the rising popularity of Graphical user interfaces, for which object-oriented programming is allegedly well-suited. An example of a closely related dynamic GUI library and OOP language can be found in the Cocoa frameworks on Mac OS X, written in Objective C, an object-oriented, dynamic messaging extension to C based on Smalltalk. OOP toolkits also enhanced the popularity of "event-driven programming" (although this concept is not limited to OOP). Some feel that association with GUI's (real or perceived) was what propelled OOP into the programming mainstream.

At ETH Zürich, Niklaus Wirth and his colleagues had also been investigating such topics as data abstraction and modular programming. Modula-2 included both, and their succeeding design, Oberon included a distinctive approach to object orientation, classes, and such. The approach is unlike Smalltalk, and very unlike C++.

Object-oriented features have been added to many existing languages during that time, including Ada, BASIC, Lisp, Fortran, Pascal, and others. Adding these features to languages that were not initially designed for them often led to problems with

compatibility and maintainability of code. "Pure" object-oriented languages, on the other hand, lacked features that many programmers had come to depend upon. To bridge this gap, many attempts have been made to create new languages based on object-oriented methods but allowing some procedural features in "safe" ways. Bertrand Meyer's Eiffel was an early and moderately successful language with those goals.

In the past decade Java has emerged in wide use partially because of its similarity to C and to C++, but perhaps more importantly because of its implementation using a virtual machine that is intended to run code unchanged on many different platforms. This last feature has made it very attractive to larger development shops with heterogeneous environments. Microsoft's .NET initiative has a similar objective and includes/supports several new languages, or variants of older ones.

More recently, a number of languages have emerged that are primarily object-oriented yet compatible with procedural methodology, such as Python and Ruby. Besides Java, probably the most commercially important recent object-oriented languages are Visual Basic .NET and C# designed for Microsoft's .NET platform.

Just as procedural programming led to refinements of techniques such as structured programming, modern object-oriented software design methods include refinements such as the use of design patterns, design by contract, and modeling languages (such as UML).

### **3.2 C# Language Fundamentals**

In C# programs, there is sufficient complexity in creating even some little program that some of the pertinent details had to be skipped over. In this part discusses the type system in C#, drawing a distinction between built-in types (int, bool, etc.) versus user-defined types (types you create as classes and interfaces). The chapter also covers programming fundamentals such as how to create and use variables and constants. It then goes on to introduce enumerations, strings, identifiers, expressions, and statements. The second part of the chapter explains and demonstrates the use of branching, using the if, switch, while, do...while, for, and foreach statements. Also discussed are operators, including the assignment, logical, relational, and mathematical operators. This is followed by an introduction to namespaces and a short tutorial on the C# precompiler.



Although C# is principally concerned with the creation and manipulation of objects, it is best to start with the fundamental building blocks: the elements from which objects are created. These include the built-in types that are an intrinsic part of the C# language as well as the syntactic elements of C#.

### **3.3 Types**

C# is a strongly typed language. In a strongly typed language you must declare the type of each object you create (e.g., integers, floats, strings, windows, buttons, etc.) and the compiler will help you prevent bugs by enforcing that only data of the right type is assigned to those objects. The type of an object signals to the compiler the size of that object (e.g., `int` indicates an object of 4 bytes) and its capabilities (e.g., buttons can be drawn, pressed, and so forth).

Like C++ and Java, C# divides types into two sets: intrinsic (built-in) types that the language offers and user-defined types that the programmer defines. C# also divides the set of types into two other categories: value types and reference types. [The principal difference between value and reference types is the manner in which their values are stored in memory. A value type holds its actual value in memory allocated on the stack (or it is allocated as part of a larger reference type object). The address of a reference type variable sits on the stack, but the actual object is stored on the heap.

C# also supports C++ style pointer types, but these are rarely used, and only when working with unmanaged code. Unmanaged code is code created outside of the .NET platform, such as COM objects.



### C# built-in value types

Type	Size(in Bytes)	.NET Type	Description
Byte	1	Byte	Unsigned (values 0-255).
Char	1	Char	Unicode characters.
Bool	1	Boolean	true or false.
sbyte	1	Sbyte	Signed (values -128 to 127).
short	2	Int16	Signed (short) (values -32,768 to 32,767).
ushort	2	UInt16	Unsigned (short) (values 0 to 65,535).
Int	4	Int32	Signed integer values between -2,147,483,647 and 2,147,483,647.
UInt	4	UInt32	Unsigned integer values between 0 and 4,294,967,295.
Float	4	Single	Floating point number. Holds the values from approximately $\pm 1.5 \times 10^{-45}$ to approximate $\pm 3.4 \times 10^{38}$ with 7 significant figures.
double	8	Double	Double-precision floating point; holds the values from approximately $\pm 5.0 \times 10^{-324}$ to approximate $\pm 1.7 \times 10^{308}$ with 15-16 significant figures.
decimal	8	Decimal	Fixed-precision up to 28 digits and the position of the decimal point. This is typically used in financial calculations. Requires the suffix "m" or "M."
Long	8	Int64	Signed integers ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
ulong	8	UInt64	Unsigned integers ranging from 0 to 0xffffffffffffff.

In addition to these primitive types, C# has two other value types: enum and struct also some other subtleties of value types such as forcing value types to act as reference types through a process known as boxing, and that value types do not "inherit."

Common escape characters

Char	Meaning
\'	Single quote
\"	Double quote
\\	Backslash
\0	Null
\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

### **3.4 C# and the .NET Framework**

The goal of C# is to provide a simple, safe, modern, object-oriented, Internet-centric, high performance language for .NET development. C# is a new language, but it draws on the lessons learned over the past three decades. In much the way that you can see in young children the features and personalities of their parents and grandparents, you can easily see in C# the influence of Java, C++, Visual Basic (VB), and other languages.

### **3.5 The .NET Platform**

When Microsoft announced C# in July 2000, its unveiling was part of a much larger event: the announcement of the .NET platform. The .NET platform is, in essence, a new development framework that provides a fresh application programming interface (API) to the services and APIs of classic Windows operating systems, especially

Windows 2000, while bringing together a number of disparate technologies that emerged from Microsoft during the late 1990s. Among the latter are COM+ component services, the ASP web development framework, a commitment to XML and object-oriented design, support for new web services protocols such as SOAP, WSDL, and UDDI, and a focus on the Internet, all integrated within the DNA architecture.

Microsoft says it is devoting 80% of its research and development budget to .NET and its associated technologies. The results of this commitment to date are impressive. For one thing, the scope of .NET is huge. The platform consists of four separate product groups:

- A set of languages, including C# and Visual Basic .NET; a set of development tools, including Visual Studio .NET; a comprehensive class library for building web services and web and Windows applications; as well as the Common Language Runtime (CLR) to execute objects built within this framework.
- A set of .NET Enterprise Servers, formerly known as SQL Server 2000, Exchange 2000, BizTalk 2000, and so on, that provide specialized functionality for relational data storage, email, B2B commerce, etc.
- An offering of commercial web services, recently announced as Project Hailstorm; for a fee, developers can use these services in building applications that require knowledge of user identity, etc.
- New .NET-enabled non-PC devices, from cell phones to game boxes.

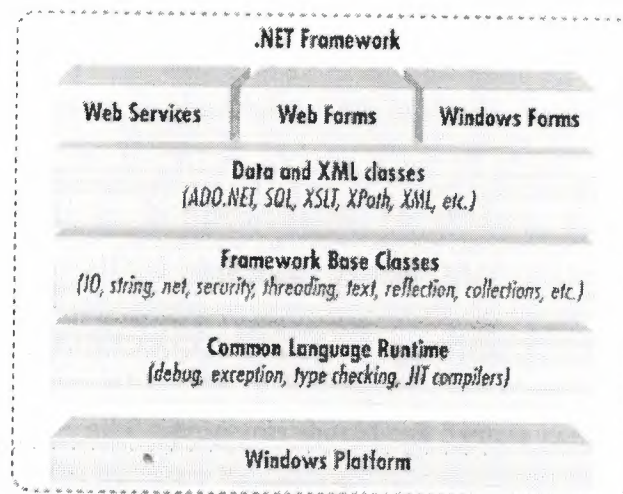
### **3.6 The .NET Framework**

Microsoft .NET supports not only language independence, but also language integration. This means that you can inherit from classes, catch exceptions, and take advantage of polymorphism across different languages. The .NET Framework makes this possible with a specification called the Common Type System (CTS) that all .NET components must obey. For example, everything in .NET is an object of a specific class that derives from the root class called System.Object. The CTS supports the general concept of classes, interfaces, delegates (which support callbacks), reference types, and value types.



Additionally, .NET includes a Common Language Specification (CLS), which provides a series of basic rules that are required for language integration. The CLS determines the minimum requirements for being a .NET language. Compilers that conform to the CLS create objects that can interoperate with one another. The entire Framework Class Library (FCL) can be used by any language that conforms to the CLS. The .NET Framework sits on top of the operating system, which can be any flavor of Windows, and consists of a number of components. Currently, the .NET Framework consists of:

- Four official languages: C#, VB .NET, Managed C++, and JScript .NET
- The Common Language Runtime (CLR), an object-oriented platform for Windows and web development that all these languages share
- A number of related class libraries, collectively known as the Framework Class Library (FCL).



**.NET Framework architecture**

### **3.7 Classes, Objects, And Types**

The essence of object-oriented programming is the creation of new types. A type represents a thing. Sometimes the thing is abstract, such as a data table or a thread; sometimes it is more tangible, such as a button in a window. A type defines the thing's general properties and behaviors.



If your program uses three instances of a button type in a window say, an “OK”, a “Cancel”, and a “Help” button each instance will share certain properties and behaviors. Each, for example, will have a size (though it might differ from that of its companions), a position (though again, it will almost certainly differ in its position from the others), and a text label (e.g., "OK", "Cancel," and "Help"). Likewise, all three buttons will have common behaviors, such as the ability to be drawn, activated, pressed, and so forth. Thus, the details might differ among the individual buttons, but they are all of the same type.

As in many object-oriented programming languages, in C# a type is defined by a class, while the individual instances of that class are known as objects. Later chapters will explain that there are other types in C# besides classes, including enums, structs, and delegates, but for now the focus is on classes.

### **3.8 Namespaces**

Namespaces are placed at the top of the .NET hierarchy. Namespaces are nothing but group of classes or types or assemblies. Each of these classes contains lot of methods. Basically, namespaces are treated as containers for all classes and are classified into several categories, based on its functionalities. For example, if you need to work with databases, you have to call the namespace System.Data. Similarly, if you are working with files you have to call System.IO namespace.

Namespaces in C# are similar to packages in Java, where we will use a statement like `java.sql.*`. Moreover, all C# programs should call System namespace. This is the root of all other namespaces in the .NET framework.

## CHAPTER -4-

### Microsoft SQL Server 2000

Microsoft SQL Server 2000 is a full-featured relational database management system (RDBMS) that offers a variety of administrative tools to ease the burdens of database development, maintenance and administration. In this article, we'll cover six of the most frequently used tools: Enterprise Manager, Query Analyzer, SQL Profiler, Service Manager, Data Transformation Services and Books Online. Let's take a brief look at each.

**4.1.1 Enterprise Manager** is the main administrative console for SQL Server installations. It provides you with a graphical "birds-eye" view of all of the SQL Server installations on your network. You can perform high-level administrative functions that affect one or more servers, schedule common maintenance tasks or create and modify the structure of individual databases.

**4.1.2 Query Analyzer** offers a quick and easy method for performing queries against any of your SQL Server databases. It's a great way to quickly pull information out of a database in response to a user request, test queries before implementing them in other applications, create/modify stored procedures and execute administrative tasks.

**4.1.3 SQL Profiler** provides a window into the inner workings of your database. You can monitor many different event types and observe database performance in real time. SQL Profiler allows you to capture and replay system "traces" that log various activities. It's a great tool for optimizing databases with performance issues or troubleshooting particular problems.

**4.1.4 Service Manager** is used to control the MSSQLServer (the main SQL Server process), MSDTC (Microsoft Distributed Transaction Coordinator) and SQLServerAgent processes. An icon for this service normally resides in the system tray of machines running SQL Server. You can use Service Manager to start, stop or pause any one of these services.

**4.1.5 Data Transformation Services (DTS)** provide an extremely flexible method for importing and exporting data between a Microsoft SQL Server

installation and a large variety of other formats. The most commonly used DTS application is the "Import and Export Data" wizard found in the SQL Server program group.

## **4.2 SQL Server Architecture**

Microsoft SQL Server 2000 data is stored in databases. The data in a database is organized into the logical components visible to users. A database is also physically implemented as two or more files on disk.

When using a database, you work primarily with the logical components such as tables, views, procedures, and users. The physical implementation of files is largely transparent. Typically, only the database administrator needs to work with the physical implementation.

Each instance of SQL Server has four system databases (master, model, tempdb, and msdb) and one or more user databases. Some organizations have only one user database, containing all the data for their organization. Some organizations have different databases for each group in their organization, and sometimes a database used by a single application. For example, an organization could have one database for sales, one for payroll, one for a document management application, and so on. Sometimes an application uses only one database; other applications may access several databases.

It is not necessary to run multiple copies of the SQL Server database engine to allow multiple users to access the databases on a server. An instance of the SQL Server is capable of handling thousands of users working in multiple databases at the same time. Each instance of SQL Server makes all databases in the instance available to all users that connect to the instance, subject to the defined security permissions.

When connecting to an instance of SQL Server, your connection is associated with a particular database on the server. This database is called the current database. You are usually connected to a database defined as your default database by the system administrator.



SQL Server 2000 allows you to detach databases from an instance of SQL Server, ~~then~~ reattach them to another instance, or even attach the database back to the same instance. If you have a SQL Server database file, you can tell SQL Server when you connect to attach that database file with a specific database name.

### **4.3 Fundamentals of SQL Server 2000 Architecture**

Microsoft SQL Server 2000 is a family of products that meet the data storage requirements of the largest data processing systems and commercial Web sites, yet at the same time can provide easy-to-use data storage services to an individual or small business.

The data storage needs of a modern corporation or government organization are very complex. Some examples are:

- \*Online Transaction Processing (OLTP) systems must be capable of handling thousands of orders placed at the same time.

- \*Increasing numbers of corporations are implementing large Web sites as a mechanism for their customers to enter orders, contact the service department, get information about products, and for many other tasks that previously required contact with employees. These sites require data storage that is secure, yet tightly integrated with the Web.

- \*Organizations are implementing off-the-shelf software packages for critical services such as human resources planning, manufacturing resources planning, and inventory control. These systems require databases capable of storing large amounts of data and supporting large numbers of users.

- \*Organizations have many users who must continue working when they do not have access to the network. Examples are mobile disconnected users, such as traveling sales representatives or regional inspectors. These users must synchronize the data on a notebook or laptop with the current data in the corporate system, disconnect from the network, record the results of their work while in the field, and then finally reconnect with the corporate network and merge the results of their fieldwork into the corporate data store.



\*Managers and marketing personnel need increasingly sophisticated analysis of trends recorded in corporate data. They need robust Online Analytical Processing (OLAP) systems easily built from OLTP data and support sophisticated data analysis.

\*Independent Software Vendors (ISVs) must be able to distribute data storage capabilities with applications targeted at individuals or small workgroups. This means the data storage mechanism must be transparent to the users who purchase the application. This requires a data storage system that can be configured by the application, and then tune itself automatically so that the users do not need to dedicate database administrators to constantly monitor and tune the application.

## **4.4 Relational Database Components**

The database component of Microsoft SQL Server 2000 is a Structured Query Language (SQL)-based, scalable, relational database with integrated Extensible Markup Language (XML) support for Internet applications. Each of the following terms describes a fundamental part of the architecture of the SQL Server 2000 database component:

### **4.4.1 Database**

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Database systems are more powerful than data files in that data is more highly organized. In a well-designed database, there are no duplicate pieces of data that the user or application must update at the same time. Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records.

When working with data files, an application must be coded to work with the specific structure of each data file. In contrast, a database contains a catalog that applications use to determine how data is organized. Generic database applications can use the catalog to present users with data from different databases dynamically, without being tied to a specific data format.

A database typically has two main parts: first, the files holding the physical database and second, the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- \*Maintaining relationships between data in the database.
- \*Ensuring that data is stored correctly, and that the rules defining data relationships are not violated.
- \*Recovering all data to a point of known consistency in case of system failures.

#### **4.4.2 Relational Database**

Although there are different ways to organize data in a database, relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

A table represents some class of objects that are important to an organization. For example, a company may have a database with a table for employees, another table for customers, and another for stores. Each table is built of columns and rows (called attributes and tuples in relational theory). Each column represents some attribute of the object represented by the table. For example, an **Employee** table would typically have columns for attributes such as first name, last name, employee ID, department, pay grade, and job title. Each row represents an instance of the object represented by the table. For example, one row in the Employee table represents the employee who has employee ID 12345.

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process called normalization, which ensures that the set of tables you define will organize your data effectively.

#### **4.4.3 Scalable**

SQL Server 2000 supports having a wide range of users access it at the same time. An instance of SQL Server 2000 includes the files that make up a set of databases and a

copy of the DBMS software. Applications running on separate computers use a SQL Server 2000 communications component to transmit commands over a network to the SQL Server 2000 instance. When an application connects to an instance of SQL Server 2000, it can reference any of the databases in that instance that the user is authorized to access. The communication component also allows communication between an instance of SQL Server 2000 and an application running on the same computer. You can run multiple instances of SQL Server 2000 on a single computer.

SQL Server 2000 is designed to support the traffic of the largest Web sites or enterprise data processing systems. Instances of SQL Server 2000 running on large, multiprocessor servers are capable of supporting connections to thousands of users at the same time. The data in SQL Server tables can be partitioned across multiple servers, so that several multiprocessor computers can cooperate to support the database processing requirements of extremely large systems. These groups of database servers are called federations.

Although SQL Server 2000 is designed to work as the data storage engine for thousands of concurrent users who connect over a network, it is also capable of working as a stand-alone database directly on the same computer as an application. The scalability and ease-of-use features of SQL Server 2000 allow it to work efficiently on a single computer without consuming too many resources or requiring administrative work by the stand-alone user. The same features allow SQL Server 2000 to dynamically acquire the resources required to support thousands of users, while minimizing database administration and tuning. The SQL Server 2000 relational database engine dynamically tunes itself to acquire or free the appropriate computer resources required to support a varying load of users accessing an instance of SQL Server 2000 at any specific time. The SQL Server 2000 relational database engine has features to prevent the logical problems that occur if a user tries to read or modify data currently used by others.

## **4.5 Structured Query Language**

To work with data in a database, you have to use a set of commands and statements (language) defined by the DBMS software. Several different languages can be



used with relational databases; the most common is SQL. The American National Standards Institute (ANSI) and the International Standards Organization (ISO) define software standards, including standards for the SQL language. SQL Server 2000 supports the Entry Level of SQL-92, the SQL standard published by ANSI and ISO in 1992. The dialect of SQL supported by Microsoft SQL Server is called Transact-SQL (T-SQL). T-SQL is the primary language used by Microsoft SQL Server applications.

#### **4.6 Database Applications and Servers**

Microsoft SQL Server 2000 is designed to work effectively as:

A central database on a server shared by many users who connect to it over a network. The number of users can range from a handful in one workgroup, to thousands of employees in a large enterprise, to hundreds of thousands of Web users.

A desktop database that services only applications running on the same desktop.

#### **4.7 Server Database Systems**

Server-based systems are constructed so that a database on a central computer, known as a server, is shared among multiple users. Users access the server through an application:

In a multitier system, such as Windows DNA, the client application logic is run in two or more locations:

A thin client is run on the user's local computer and is focused on displaying results to the user.

The business logic is located in server applications running on a server. Thin clients request functions from the server application, which is itself a multithreaded application capable of working with many concurrent users. The server application is the one that opens connections to the database server. The server application can be running on the same server as the database, or it can connect across the network to a separate server operating as a database server. In complex systems, the business logic may be implemented in several interconnected server applications, or in multiple layers of server applications.



This is a typical scenario for an Internet application. For example, a multithreaded server application can run on a Microsoft Internet Information Services (IIS) server and service thousands of thin clients running on the Internet or an intranet. The server application uses a pool of connections to communicate with one or more instances of SQL Server 2000. The instances of SQL Server 2000 can be on the same computer as IIS, or they can be on separate servers in the network.

In a two-tier client/server system, users run an application on their local computer, known as a client application, that connects over a network to an instance of SQL Server 2000 running on a server computer. The client application runs both business logic and the code to display output to the user, so this is sometimes referred to as a thick client.

#### **4.8 Advantages of Server Database System**

Having data stored and managed in a central location offers several advantages:

Each data item is stored in a central location where all users can work with it.

Separate copies of the item are not stored on each client, which eliminates problems with users having to ensure they are all working with the same information. Their system does not need to ensure that all copies of the data are updated with the current values, because there is only one copy in the central location.

Business and security rules can be defined one time on the server and enforced equally among all users.

Rule enforcement can be done in a database through the use of constraints, stored procedures, and triggers. Rules can also be enforced in a server application, since these applications are also central resources accessed by many thin clients.

A relational database server optimizes network traffic by returning only the data an application needs.

For example, if an application working with a file server needs to display a list of the names of sales representatives in Oregon, it must retrieve the entire employee file. If the application is working with a relational database server, it sends this command:

```
SELECT first_name, last_name  
FROM employees  
WHERE emp_title = 'Sales Representative'
```

AND emp\_state = 'OR'

The relational database sends back only the names of the sales representatives in Oregon, not all of the information about all employees.

Hardware costs can be minimized.

Because the data is not stored on each client, clients do not have to dedicate disk space to storing data. The clients also do not need the processing capacity to manage data locally, and the server does not need to dedicate processing power to displaying data.

The server can be configured to optimize the disk I/O capacities needed to retrieve data, and clients can be configured to optimize the formatting and display of data retrieved from the server.

The server can be stored in a relatively secure location and equipped with devices such as an Uninterruptable Power Supply more economically than fully protecting each client.

Maintenance tasks such as backing up and restoring data are simplified because they can focus on the central server.

#### **4.9 Advantages of SQL Server 2000 as a Database Server**

Microsoft SQL Server 2000 is capable of supplying the database services needed by extremely large systems. Large servers may have thousands of users connected to an instance of SQL Server 2000 at the same time. SQL Server 2000 has full protection for these environments, with safeguards that prevent problems, such as having multiple users trying to update the same piece of data at the same time. SQL Server 2000 also allocates the available resources effectively, such as memory, network bandwidth, and disk I/O, among the multiple users.

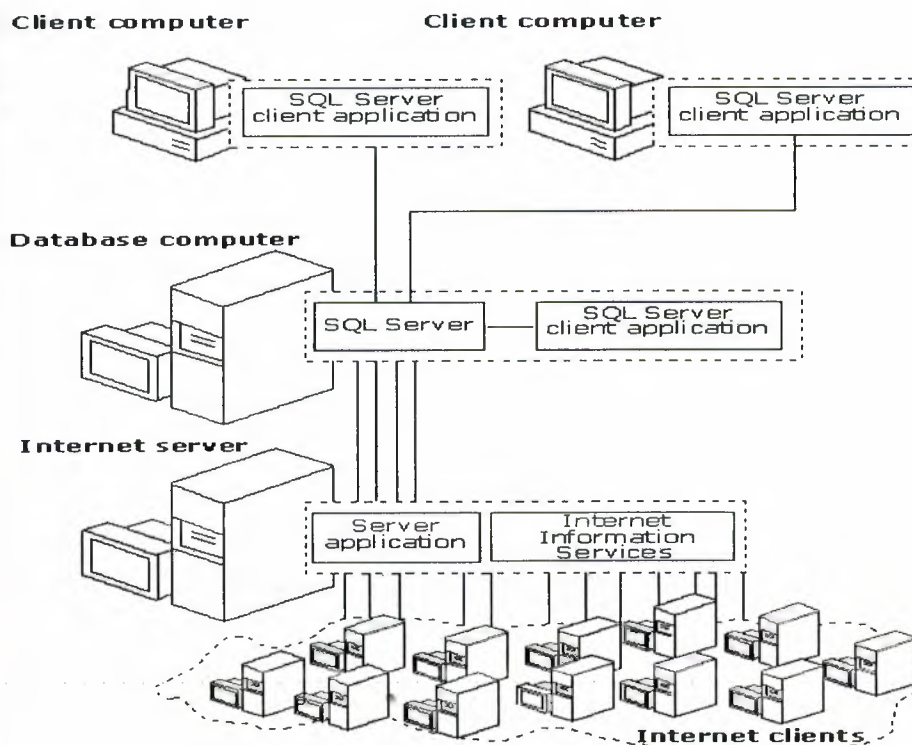
Extremely large Internet sites can partition their data across multiple servers, spreading the processing load across many computers, and allowing the site to serve thousands of concurrent users.

Multiple instances of SQL Server 2000 can be run on a single computer. For example, an organization that provides database services to many other organizations can run a separate instance of SQL Server 2000 for each customer organization, all on one

computer. This isolates the data for each customer organization, while allowing the service organization to reduce costs by only having to administer one server computer.

SQL Server 2000 applications can run on the same computer as SQL Server 2000. The application connects to SQL Server 2000 using Windows Interprocess Communications (IPC) components, such as shared memory, instead of a network. This allows SQL Server 2000 to be used on small systems where an application must store its data locally.

The illustration shows an instance of SQL Server 2000 operating as the database server for both a large Web site and a legacy client/server system.



The largest Web sites and enterprise-level data processing systems often generate more database processing than can be supported on a single computer. In these large systems, the database services are supplied by a group of database servers that form a database services tier. SQL Server 2000 does not support a load-balancing form of clustering for building a database services tier, but it does support a mechanism that can



be used to partition data across a group of autonomous servers. Although each server is administered individually, the servers cooperate to spread the database-processing load across the group. A group of autonomous servers that share a workload is called a federation of servers.

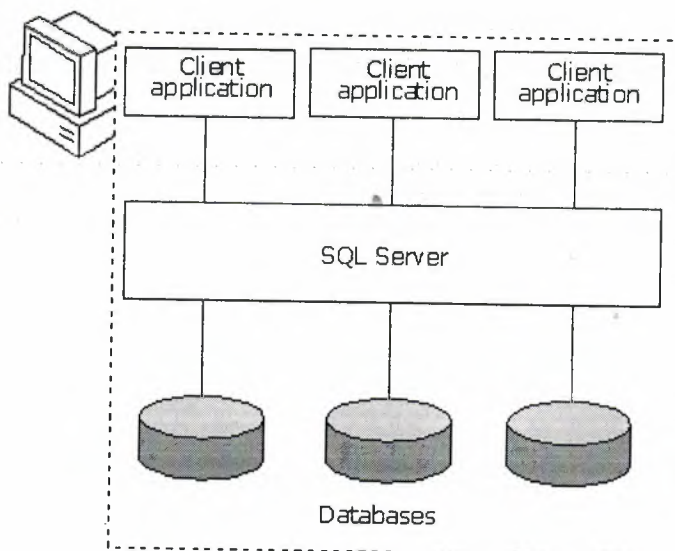
#### 4.10 Desktop Database Systems

Although SQL Server 2000 works effectively as a powerful database server, the same database engine can also be used in applications that need stand-alone databases stored locally on the client. SQL Server 2000 can configure itself dynamically to run efficiently with the resources available on a client desktop or laptop computer, without the need to dedicate a database administrator to each client. Application vendors can also embed SQL Server 2000 as the data storage component of their applications.

When clients use local SQL Server 2000 databases, applications connect to local instances of the database engine in much the same way they connect across the network to a database engine running on a remote server. The primary difference is that local connections are made through local IPCs such as shared memory, and remote connections must go through a network.

The illustration shows using SQL Server 2000 in a desktop database system.

Desktop Computer





## **CHAPTER -5-**

### **OVERVIEW TO CMC CAR GALLERY PROJECT**

In this chapter we will analyze the “CMC CAR GALLERY” web page. And also this chapter shows you how to build a web application (web form) for car galleries step by step, using all the programming languages, database system and techniques which are covered in the previous chapters. This web application case study will take you through all the important factors to be considered when creating a web application. This web service is created in two stages. These stages are database stage and web page stage. In database stage as you know MS SQL Server 2000 is used in project, the second stage, web page, the coding languages which are ASP.NET and C# languages is used.

While developing a complete web page for second hand car gallery programme tried to cover real world rules, which are necessary to make connection between administrators and the users, to avoid losing members and visitors.

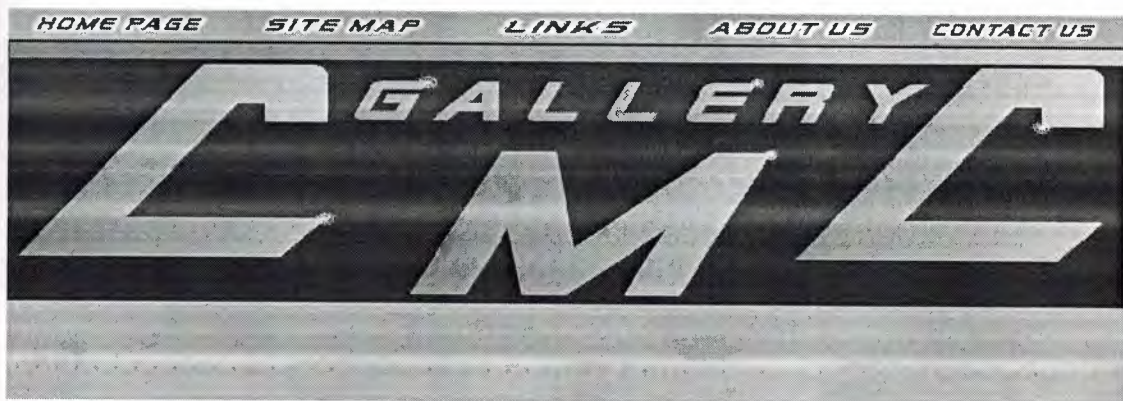
In this project as written before there are two stages, now start to overview with the database then web page design. But before starting overview next stage will give some information about web page structure in two parts, which are user side and administrator side.

#### **5.1 Web Page Structure of User Side:**

In user side, the user reach home page of the CMC gallery at the beginning. Then they will choose their options to do their operation. But it is important the user who has not an account of the “CMC Gallery” web page their operations limited, to use all the contents of the web site (user side) they have to create an account. Let’s start to investigate the web page as user step by step.

### 5.1.1 Top Side of The Pages

In every page you will see the same top design as the figure. In this design there is main logo of the web site and there is five image buttons which are “Home Page”, “Site Map”, “Links”, “About us”, and “Contact Us”. These image buttons forwards you to related pages. Orderly index.aspx, sitemap.aspx, links.aspx, aboutus.aspx and contactus.aspx.



### 5.1.2 Home Page

In the home page which is our main page, there is seven chooses to reach the related pages. Let me define them in orderly.

5.1.2.1 “CAR” button refers to search page where user can easily search cars.

5.1.2.2 “COMMERCIAL” button refers to commercial search page.

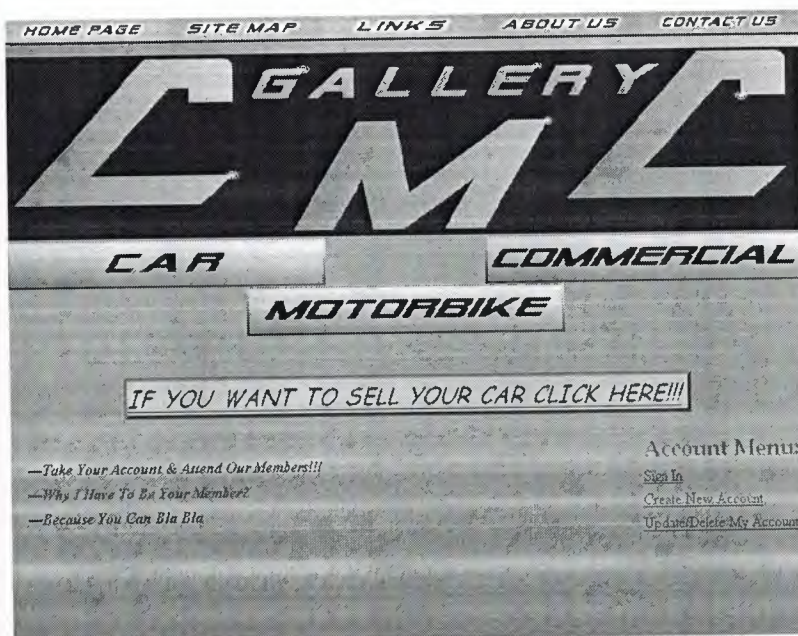
5.1.2.3 “MOTORBIKE” button forward users to commercial vehicle search page.

5.1.2.4 “IF YOU WANT TO SELL YOUR CAR CLICK HERE!!!!” image button reach to sign in page because just the submitted users can sell their vehicle on the page.

5.1.2.5 “Sign In” link directly send the users to the sign in page.

5.1.2.6 “Create New Account” link sends to user to the create account page to create new account.

5.1.2.7 “Update/Delete My Account” forwards users to the sign in page.



### 5.1.3 Search Pages

In web site there are three different search pages for each vehicle types which are car, motorbike and commercial. Each page includes two drop down lists, two text boxes and three buttons. These tools are used for following things.

5.1.3.1 First drop down list includes the vehicle makes user can select one of them.

5.1.3.2 Second drop down list is for models of the defined make.



5.1.3.3 The text box which is located on the other is to define minimum price of

5.1.3.4 The text box which is located under the previous text box is to define the maximum price for searching.

5.1.3.5 The button which is located left side-bottom of the page is starting the searching by the defined make and model.

5.1.3.6 The right-bottom side button starts searching operation between the selected values.

5.1.3.7 user can also use the button which is putted the bottom-center of the page, to start searching both make, model and price.

The screenshot displays the CMC Gallery website's search interface. At the top, a navigation bar contains links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below this is a large, stylized logo for 'CMC GALLERY'. The main section is titled 'QUICK MOTORBIKE SEARCH'. It features two dropdown menus for 'Make' and 'Model'. To the right, there are two text input fields for 'Minimum Price' and 'Maximum Price', with a 'BETWEEN' label between them. At the bottom, there are three buttons: 'Search By Make&Model' on the left, 'Search By Price' on the right, and 'Search By Make,Model And Price' in the center.

#### 5.1.4 Sign In Page

Sign in page includes two text boxes, two buttons and a link. This page is an interface page to reach user operation page. The tools operations are:



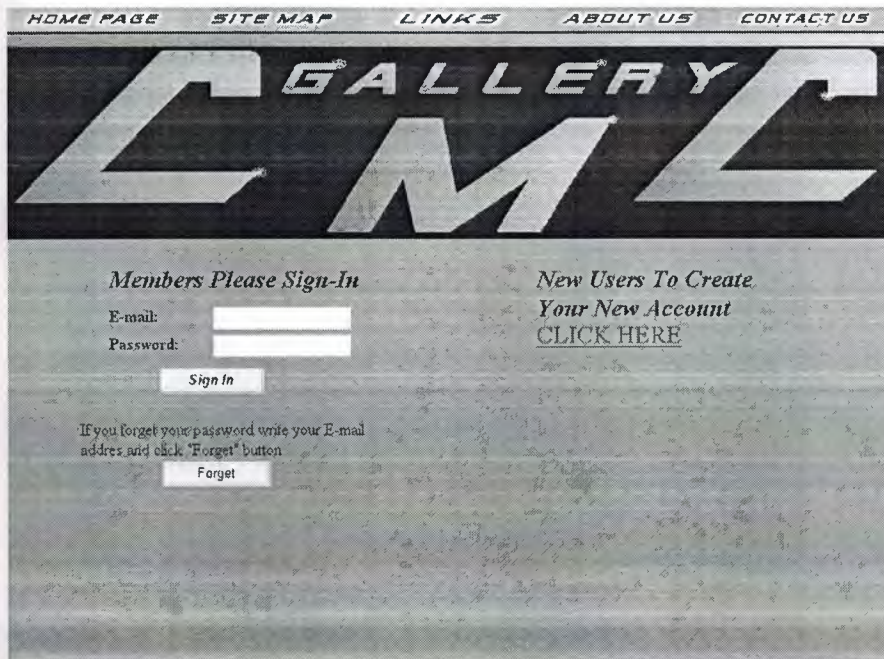
5.1.4.1 First text box for member E-mail address.

5.1.4.2 Second text box is for user password.

5.1.4.3 "Sign In" button for stepping to user operations page.

5.1.4.4 "Forget" button for reaching the forget password page where user constitute new password.

5.1.4.5 "CLICK HERE" link forwards user to create new account page.

The image shows a web page for 'CMC GALLERY'. At the top, there is a navigation bar with links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below this is a large, stylized 'CMC GALLERY' logo. The main content area is divided into two sections. On the left, under the heading 'Members Please Sign-In', there are two text input fields labeled 'E-mail:' and 'Password:', followed by a 'Sign In' button. Below these fields is a link that says 'If you forget your password write your E-mail address and click "Forget" button' with a 'Forget' button underneath. On the right, under the heading 'New Users To Create Your New Account', there is a 'CLICK HERE' link.

### 5.1.5 Forget Password Page

In forget password page user who could not remember the password, creates new password with answering their secret question. This page contains one special label which is different from other labels, three text boxes and a button.

5.1.5.1 Special label shows the question which user designated while creating an account.

5.1.5.2 First check box is for user answer.

5.1.5.3 Second text box is for new password.

5.1.5.4 Last text box for confirming the new password.

5.1.5.5 Button is for submitting the new password. This button is also refers to user operations page if the conditions available.

HOME PAGE SITE MAP LINKS ABOUT US CONTACT US

# CMC GALLERY

## FORGET YOUR PASSWORD

Your Secret Question Was: What is your favourite vehicle

Please Type Your Answer:

Please Enter Your New Password:

Please Confirm Your Password:

### 5.1.6 Create New User Page

Users can easily create an account on this page with filling the necessary information. This page also contains ten text boxes, one drop down list and a button.

5.1.6.1 First text box is for user E-mail address.

5.1.6.2 Second text box is for new password.

5.1.6.3 Third text box for confirming the new password.

5.1.6.4 Fourth text box for user name.

5.1.6.5 Fifth text box for user surname.

5.1.6.6 Sixth text box for user phone number.

5.1.6.7 Seventh text box which is the smallest text box, for user mobile phone code.



5.1.6.8 Eighth text box for user phone number instead of whole number, just last seven digits of the number.

5.1.6.9 Ninth text box for user fax number.

5.1.6.10 The last text box is for secret question answer.

5.1.6.11 Drop down list is for choosing the secret question.

5.1.6.12 Button is used for submitting the form.

HOME PAGE   SITE MAP   LINKS   ABOUT US   CONTACT US

# CMC GALLERY

## MEMBER ACCOUNT FORM

"" means field is required

E-Mail:

Password:

Confirm:

Name:

Surname:

Phone No.:  (Ex. 228\*\*\*\*)

GSM No.+905:  (Ex. 33- 877\*\*\*\*)

Fax:  (Ex. 227\*\*\*\*)

Secret Question:

Answer:

### 5.1.7 User Operations Page

On User Operations page user can choose their operation with buttons, they have several operations for them to do. Page supports eight buttons for users.

5.1.7.1 “Sell a Vehicle” button refers to sell vehicle page.

5.1.7.2 “Delete My Account” button sends the user to the update or delete account page.

5.1.7.3 “Send Message To Admin” button directly connects the users to the send message page.

5.1.7.4 “Update/Delete Vehicle Info” button moves to user to the update or delete your vehicle page.

5.1.7.5 “Update Personal Info” button takes to user to the update info page.

5.1.7.6 “CAR” button refers to search page where user can easily search cars.

5.1.7.7 “COMMERCIAL” button refers to commercial search page.

5.1.7.8 “MOTORBIKE” button forward users to commercial vehicle search page.



### 5.1.8 Sell a Vehicle

This page collects information about the vehicles which are wanted to sell. Page includes one radio button group, three drop down lists, nine text boxes, two buttons and a check box group.

5.1.8.1 Radio button group include three types of vehicles user to choose one of them which are orderly car, motorbike and commercial.



5.1.8.2 Drop down lists allows users to select their vehicle make, model, and fuel type.

5.1.8.3 Small text boxes collect specified information about the vehicle.

5.1.8.4 The big (Multi Line) text box stores more statements about the vehicle.

5.1.8.5 "Add Image" button allows users to upload the vehicle pictures.

5.1.8.6 The check box groups collect the vehicles extras.

5.1.8.7 "Add To Gallery" button accept the vehicle and stores it in the site database.

The screenshot shows a web form titled "CMC GALLERY" with a navigation bar at the top containing links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below the title is a section "SELL YOUR" with radio buttons for "CAR", "MOTORBIKE", and "COMMERCIAL". The form contains several input fields: "Make" (a dropdown menu), "Variant" (a text box), "Color" (a text box), "KM Age" (a text box), and "Information" (a text box). Below these are "Model" (a dropdown menu), "Engine" (a text box), "Registry Year" (a text box), and "Price" (a text box). Further down are "Fuel Type" (a dropdown menu), "Gear Type" (a text box), and "Registry Plate" (a text box). There is an "Add Image" button. At the bottom, there is a section "Extras" with a grid of checkboxes for various features: Alloy wheels, Data airbag, Power steering, Central locking, Electric sunroof, Electric windows, Electric mirrors, Electric seats, Parking sensors, Air conditioning, Climate control, Driver airbag, ABS, CB system, CD autochanger, Radio cassette, Alarm, Trunk buzzer, Heated windscreen, Mobile kit, Fog lamps, Tinted glass, Remote key locking, and Spoiler. A "Submit" button is located at the bottom center of the form.

### 5.1.9 Send Message page

Users can send message to administrator on this page. This page contains three text boxes, and one button.

5.1.9.1 First text box for E-mail address of the user.

5.1.9.2 Second text box for subject of the message.

5.1.9.3 Last text box which is multi line text box for message.

5.1.9.4 “Send My Message” button for sending the message to the administrator.

The screenshot shows a web page with a navigation bar at the top containing links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below the navigation bar is a large logo that reads 'GALLERY CMC'. The main content area is titled 'SEND MESSAGE'. It contains three input fields: 'E-Mail:' with a single-line text box, 'Subject:' with a single-line text box, and 'Message:' with a multi-line text area. At the bottom right of the form is a button labeled 'Send My Message'.

### 5.1.10 Update or Delete User Page

This page includes two operations for users who want to update or delete their account. Page includes ten text boxes, one drop down list and two buttons.

5.1.10.1 First text box is for user who wants to update E-mail address.

5.1.10.2 Second text box is for new password.

5.1.10.3 Third text box for confirming the new password.

5.1.10.4 Fourth text box for user who wants to update name.

5.1.10.5 Fifth text box for user who wants to update surname.

5.1.10.6 Sixth text box for user who wants to update phone number.

5.1.10.7 Seventh text box which is the smallest text box, for user who wants to update mobile phone code.

5.1.10.8 Eighth text box for user who wants to update seven digits phone number.

5.1.10.9 Ninth text box for user who wants to update fax number.

5.1.10.10 The last text box is for changing secret question answer.



5.1.10.11 Drop down list is for changing the secret question.

5.1.10.12 "Update My Account" button is used for updating the information.

5.1.10.13 "Delete My Account" button is used for deleting the account.

The screenshot shows a web page with a navigation bar at the top containing links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below the navigation bar is a large graphic with the text 'GALLERY' and 'CMC' in a stylized font. The main content area is titled 'UPDATE or DELETE ACCOUNT'. It contains a form with the following fields: E-Mail, Password, Confirm, Name, Surname, Phone No, GSM No. +905, Fax, Secret Question, and Answer. To the right of the Phone No, GSM No. +905, and Fax fields are example numbers: (Ex: 228\*\*\*\*), (Ex: 33- 877\*\*\*\*), and (Ex: 227\*\*\*\*) respectively. A note '\* \* \* means field is required' is present. At the bottom of the form are two buttons: 'UPDATE MY ACCOUNT' and 'DELETE MY ACCOUNT'.

### 5.1.11 Update or Delete Vehicle Page

Users can do two types of operation on this page first one is updating and second one is deleting their vehicle information. In this page there are one radio button group, three drop down lists, nine text boxes, two buttons and a check box group.

5.1.11.1 Radio button group include three types of vehicles user to change one of them.

5.1.11.2 Drop down lists allows users to change their vehicle make, model, and fuel type.

5.1.11.3 Small text boxes update specified information about the vehicle.

5.1.11.4 The big (Multi Line) text box change information statements about the vehicle.

5.1.11.5 “Add Image” button allows users to update the vehicle pictures.

5.1.11.6 The check box group updates the vehicles extras.

5.1.11.7 “Update” button accept the changes and stores new data in the site database.

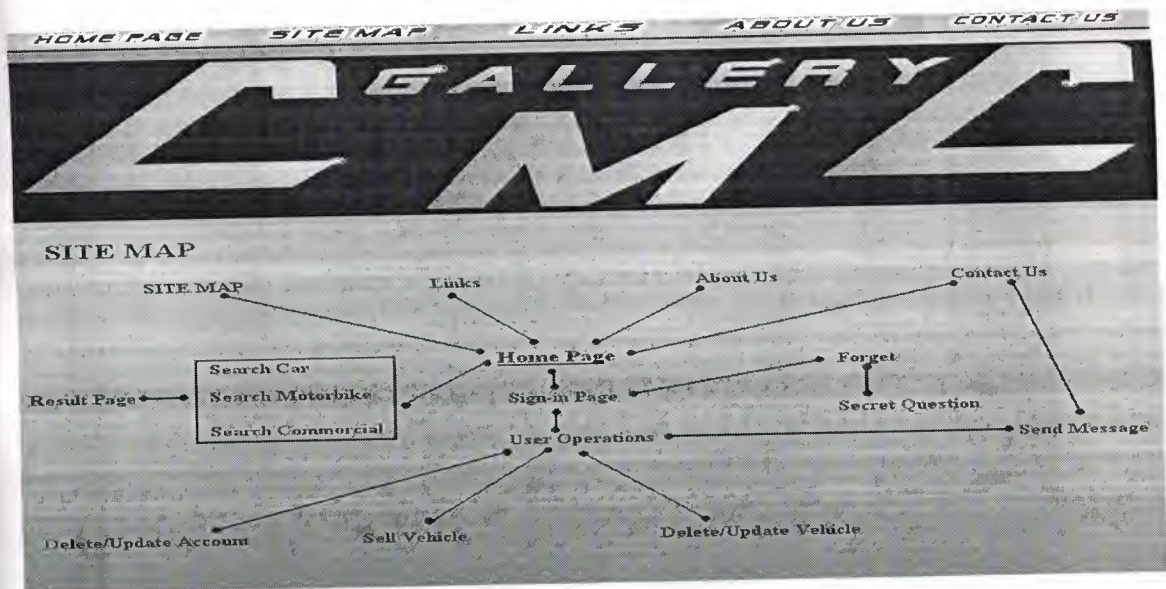
5.1.11.8 “Delete” button removes all information about the vehicle from the site database.

The screenshot shows the 'CMC GALLERY' website interface. At the top, there is a navigation bar with links: HOME PAGE, SITE MAP, LINKS, ABOUT US, and CONTACT US. Below this is a large banner with the text 'CMC GALLERY'. Under the banner, there is a section titled 'SELL YOUR' followed by radio buttons for 'CAR', 'MOTORBIKE', and 'COMMERCIAL'. The main form is divided into several sections: 'Make' (dropdown), 'Variant' (text), 'Color' (text), 'KM Age' (text), and 'Information' (text). Below these are 'Model' (dropdown), 'Engine' (text), 'Registry Year' (text), and 'Price' (text). Further down are 'Fuel Type' (dropdown), 'Gear Type' (text), and 'Registry Plate' (text). An 'Add Image' button is located to the right of the 'Registry Plate' field. The 'Extras' section contains a grid of checkboxes for various vehicle features: Alloy wheels, Dual airbag, Power steering, Central locking, Electric sunroof, Electric windows, Electric mirrors, Electric seats, Parking sensors, Air conditioning, Climate control, Driver airbag, ABS, CD system, CD autochanger, Radio cassette, Alarm, Immobiliser, Heated windscreen, Mobile kit, Fog lamps, Tinted glass, Remote key locking, and Spoiler. At the bottom of the form, there are two buttons: 'Update' and 'Delete'.

## 5.1.12 Site Map

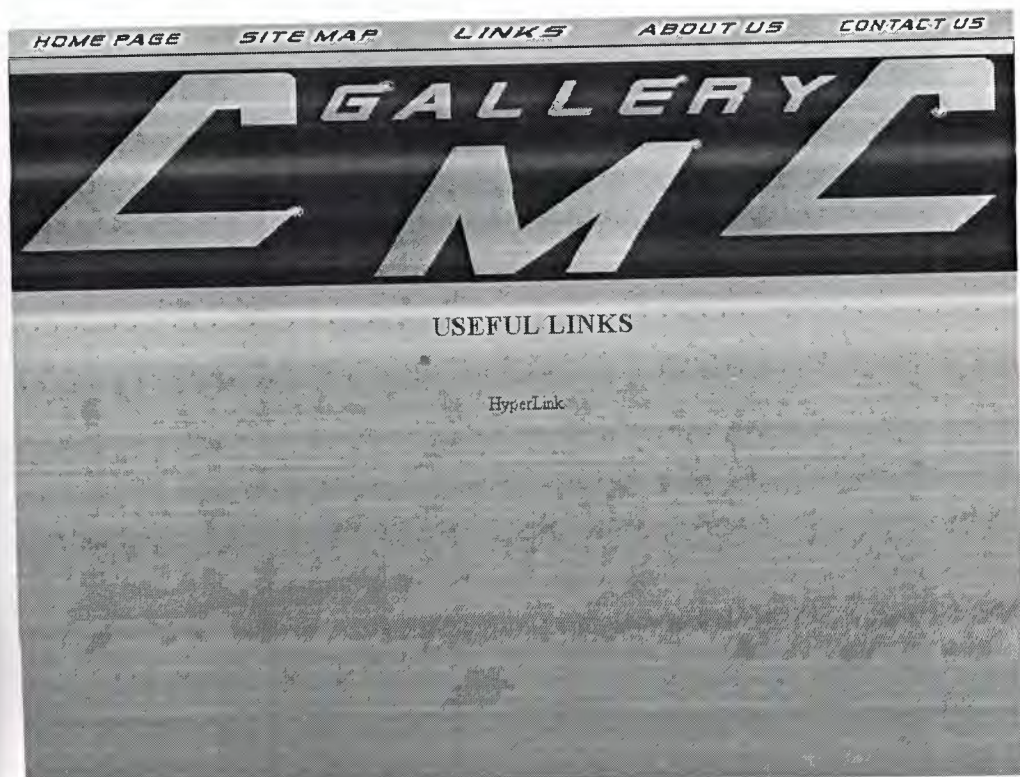
In site map page it shows hierarchy of the page and also obtains all the links of the available pages.





### 5.1.13 Links Page

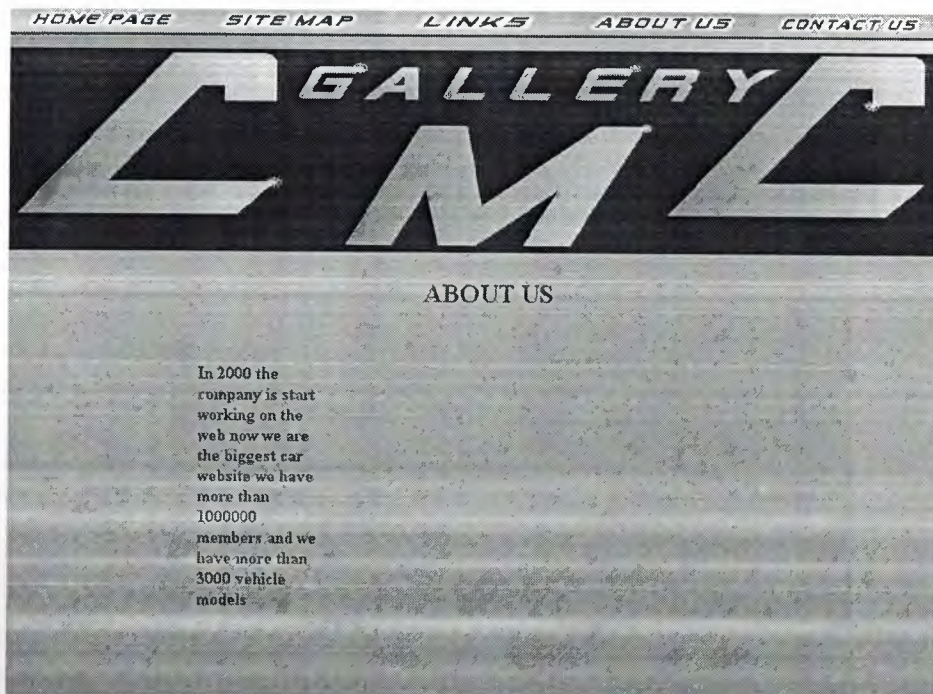
Links Page includes some useful links which are added by administrator.





### 5.1.14 About Us Page

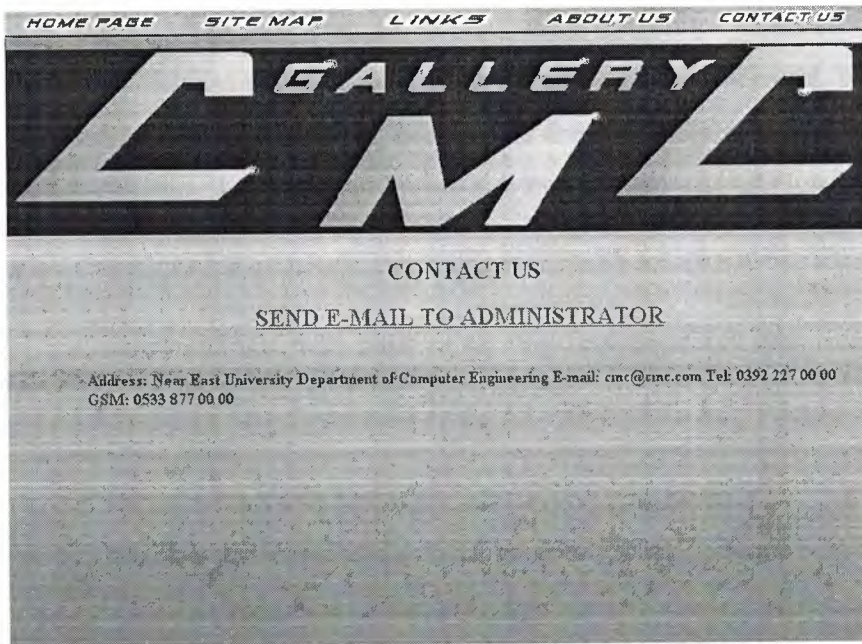
This page gives information to the visitors about the company. The text is written by the administrator.



### 5.1.15 Contact Us Page

This page give information to the user about how the user can reach the company and also page is include one link

5.1.15.1 Link forwards user to send message page.



#### 5.1.16 Result Page

This page shows the result of the vehicle search which depends on user criteria. Page contains tree button and a data grid.

5.1.16.1 “CAR” button refers to search page of car for new car search.

5.1.16.2 “COMMERCIAL” button refers to commercial search page for new search.

5.1.16.3 “MOTORBIKE” button forward users to commercial vehicle search page for searching again.

5.1.16.4 Data grid shows the result of the related search.

#### 5.2 Web Page Structure of Administrator Side:

In administrator side, the administrator reach administration page of the CMC gallery at the beginning. Then they must enter the user name and password. Because it is very important the person who has not an administrator of the “CMC Gallery” web page



they are not allow to operate on administrator operations page. To use all the contents of the web site (administrator side) they have to create an account, but they have to create new account on the database, because on the web page it is very risky without a strong security system. Let's start to investigate the web page as administrator step by step.

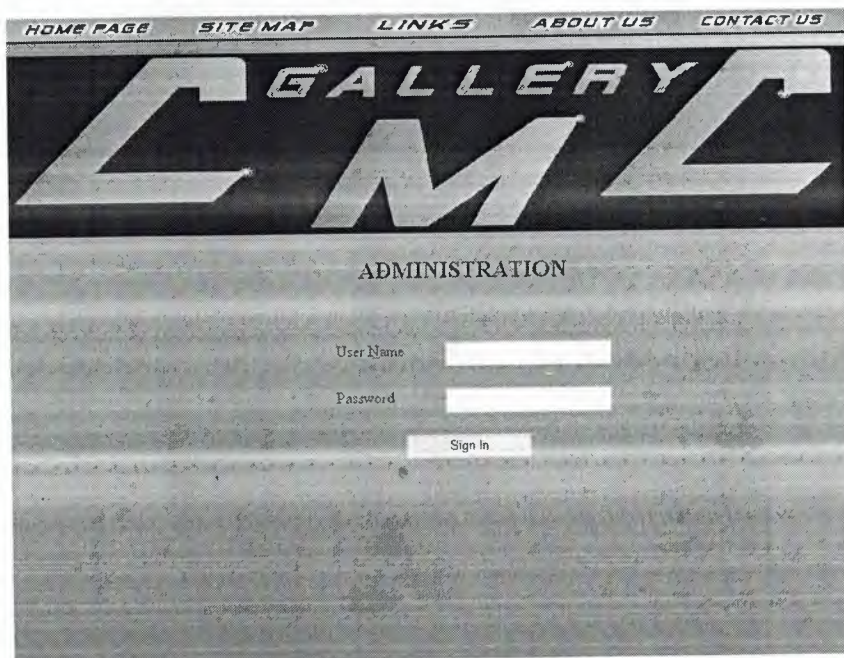
### 5.2.1 Administrator Sign In Page

Administrator sign page is an interface page for administrator operation page. This page contains two text box and a button.

5.1.17.1 First text box is for administrator user name.

5.1.17.2 Second text box is for administrator password.

5.1.17.3 "Sign In" button is for passing the administrator operation page if conditions are available.





### 5.2.2 Administrator Operation Page

Administrator can operate everything about the web site on this page with using buttons. Because of on this page everything is invisible without six buttons which are placed under the company logo. With clicking the buttons administrator easily view the entries and also remove user, vehicle and messages from the site database. There is another thing about the contact us, about us, and links. Administrator also can change or delete these statements and links.

m_id	email	subject	message
1	den@den.den	dene	denerne 1.

### 5.3 Creating Database & Database Schema

While creating the database for car gallery we should well analyze the elements of our database such as primary key and foreign keys. Also we have to arrange the relations between the tables in CMC Car Gallery web sites database ver well. CMC Car Gallery database supports basic database tables in two groups; first group is related with users and second group is related with administrators. In the first group, there are nine tables: *info\_user*, *sec\_quest*, *info\_vehicle*, *vehicles*, *makes*, *models*, *fuels*, *gears* and *extras*. In the second group there are four tables: *admin*, *messages*, *links* and *catable*. The information of the corresponding tables are as follows:

#### First group:

<i>info_user</i>	A table contains user information
<i>sec_quest</i>	A table contains secret questions (related with <i>info_user</i> )
<i>info_vehicle</i>	A table contains vehicle information.
<i>vehicles</i>	A table contains vehicle types (related with <i>info_vehicle</i> )
<i>makes</i>	A table contains vehicle makes (related with <i>info_vehicle</i> & <i>vehicle</i> )
<i>models</i>	A table contain vehicle models (related with <i>info_vehicle</i> & <i>makes</i> )
<i>fuels</i>	A table contains fuel types (related with <i>info_vehicle</i> )
<i>gears</i>	A table contains gear types (related with <i>info_vehicle</i> )
<i>extras</i>	A table contains vehicle features (related with <i>info_vehicle</i> )

#### Second group:

<i>admin</i>	A table contains administrators information
<i>messages</i>	A table contains received messages from visitors
<i>links</i>	A table contains links shown in “links” page
<i>catable</i>	A table contains texts which are shown in “contact us” and “about us” pages

After deciding our tables now we arrange their column names, data types and lengths orderly. In the following parts you will see that how the database arranged the tables with table pictures and also you will get information about the columns' characteristics.

### 5.3.1 Tables

#### 1) *info\_user* table

Design Table 'info\_user' in 'galeri' on '(LOCAL)

Column Name	Data Type	Length	Allow Nulls
id_no	int	4	
email	varchar	50	
password	varchar	50	
name	varchar	50	✓
surname	varchar	50	✓
phone	varchar	8	✓
gsm_code	varchar	3	✓
gsm	varchar	8	✓
fax	varchar	8	✓
secret_q	smallint	2	
secret_answer	varchar	50	

Columns

Description

Default Value

Precision 10

Scale 0

Identity Yes (Not For Replication)

Identity Seed 1

Identity Increment 1

Is RowGuid No

Formula

Collation

In *info\_user* table, *id\_no* is the primary key and *secret\_q* is the foreign key of the *sec\_quest* table. This table stores the user information which are shown in the following table with examples.



Column Name	Refers	Example
email	User E-mail address	user@cmc.com
password	User password	pass123
name	User name	james
surname	User surname	brown
phone	User phone number	2270000
gsm_code	User GSM no. code	533
gsm	User GSM number	8770000
fax	User fax number	2271111
secret_q	Selected secret question id from sec_quest table	5
secret_answer	User answer for selected secret question	blue

## 2) sec\_quest table

Design Table 'sec\_quest' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Null:
secret_q	smallint	2	
secret_question	varchar	50	
	char	10	

Columns

Description

Default Value

Precision

Scale

Identity

Identity Seed

Identity Increment

Is RowGuid

Formula

Collation

<database default>

In *sec\_quest* table, *sec\_q* is the primary key. This table stores the predefined secret question for user to choose one of them. The details are shown in the following table with examples.

Column Name	Refers	Example
secret_q	Secret question id number	5
secret_question	Secret question	What is your favorite color

### 3) *info\_vehicle* table

Design Table 'info\_vehicle' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Nulls
vid_no	int	4	
id_no	int	4	
vehicle_type	tinyint	1	
make	int	4	
model	int	4	
variant	varchar	50	
engine	varchar	50	
gear_type	smallint	2	
reg_year	varchar	50	
color	varchar	50	✓
fuel	tinyint	1	
reg_plate	char	10	✓
km_age	bigint	8	✓
price	bigint	8	
info	varchar	500	✓
photo	image	16	✓

Columns

Description

Default Value

Precision: 0

Scale: 0

Identity: No

Identity Seed


Identity Increment

Is Rowid: No

Formula

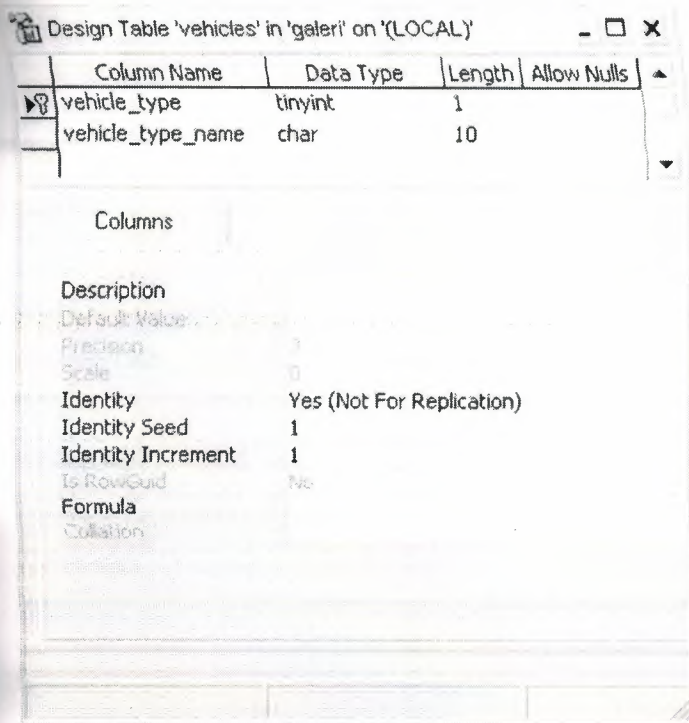
Collation

In *info\_vehicle* table *vid\_no* is the primary key and *id\_no*, *vehicle\_type*, *make*, *model*, *gear\_type* and *fuel* are the foreign key of orderly the *info\_user*, *vehicle*, *makes*, *models*, *gears* and *fuels* tables. This table stores the vehicle information which are shown in the following table with examples.

Column Name	Refers	Example
vid_no	Vehicle id no	1
id_no	Vehicle owner (user) id no	2
vehicle_tyoe	Vehicle type id	1
make	Make id no	28
model	Model id no	145
variant	Vehicle variant type	Sport
engine	Engine type	2000
gear_type	Gear id no	1
reg_year	Registration year	2004
color	Vehicle color	Metallic grey
fuel	Fuel id no	1
reg_plate	Vehicle registry plate	RT 300
km_age	Vehicle km age	28000
price	Vehicle price (YTL)	45000
info	Extra information	I can exchange with BMW
photo	Vehicle photo	



#### 4) vehicles table



In *vehicles* table *vehicle\_type* is the primary key. This table stores the predefined *vehicle* types such as car, commercial and motorbike for user to choose one of them. The details are shown in the following table with examples.

Column Name	Refers	Example
vehicle_type	Vehicle type id no	2
vehicle_type_name	Type of the vehicle	motorbike

### 5) makes table

Design Table 'makes' in 'galeri' on '(LOCAL)'

	Column Name	Data Type	Length	Allow Nulls
	make	int	4	
	make_name	varchar	50	
	vehicle_type	tinyint	1	

Columns

Description

Default Value

Precision 10

Scale 0

Identity Yes (Not For Replication)

Identity Seed 1

Identity Increment 1

Is RowGuid No

Formula



Collation

In the *makes* table, *make* is the primary key. This table stores the predefined vehicle makes for user to select one of them. The properties are shown in the following table with examples.

Column Name	Refers	Example
make	Vehicle's make id	98
make_name	Vehicle' make	Ducati
vehicle_type	Id number of the vehicle type	2

## 6) *models* table

Design Table 'models' in 'galeri' on '(LOCAL)'

	Column Name	Data Type	Length	Allow Nulls
	model	int	4	
	model_name	varchar	50	
	make	int	4	<input type="checkbox"/>

Columns

Description

Default Value

Precision

Scale

Identity

Identity Seed

Identity Increment

Is RowGuid

Formula

Collation

In the *models* table *model* is the primary key. This table stores the predefined vehicle's models for user to select one of them. The details are shown in the following table with examples.

Column Name	Refers	Example
model	Model id no	75
model_name	Name of the vehicle' model	Broadway
make	Id no of the vehicle's make	19



## 7) *fuels* table

Design Table 'fuels' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Nulls
fuel	tinyint	1	
fuel_type	char	10	✓

Columns

Description

Default Value

Precision

Scale

Identity Yes (Not For Replication)

Identity Seed 1

Identity Increment 1

Is RowGuid No

Formula

Collation

In the *fuels* table *fuel* is the primary key. This table stores the predefined vehicle's fuel types for user to select one of them. The details are shown in the following table with examples.

Column Name	Refers	Example
fuel	Fuel id no	2
fuel_type	Type of the fuel	Diesel

## 8) *gears* table

Design Table 'gears' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Nulls
gear_type	smallint	2	
gear	varchar	50	✓

Columns

Description

Nullable Value

Precision

Scale

Identity

Identity Seed

Identity Increment

Is RowGuid

Formula

Collation

In the *gears* table *gear\_type* is the primary key. This table stores the predefined gear types for user to select one of them. The details are shown in the following table with examples.

Column Name	Refers	Example
gear_type	Id no of the gear type	1
gear	Gear type of the vehicle	Manual

## 9) extras table

Design Table 'extras' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Nulls
vid_no	int	4	
[Alloy wheels]	bit	1	✓
[Dual airbag]	bit	1	✓
[Power steering]	bit	1	✓
[Central locking]	bit	1	✓
[Electric sunroof]	bit	1	✓
[Electric windows]	bit	1	✓
[Electric mirrors]	bit	1	✓
[Electric seats]	bit	1	✓
[Parking sensors]	bit	1	✓
[Air conditioning]	bit	1	✓
[Climate control]	bit	1	✓
[Driver airbag]	bit	1	✓
ABS	bit	1	✓
[CD system]	bit	1	✓
[CD autochanger]	bit	1	✓
[Radio cassette]	bit	1	✓
Alarm	bit	1	✓
Immobiliser	bit	1	✓
[Heated Windscreen]	bit	1	✓
[Mobile kit]	bit	1	✓
[Fog lamps]	bit	1	✓
[Tinted glass]	bit	1	✓
[Remote key locking]	bit	1	✓
Spoiler	bit	1	✓

Columns

Description

Default Value

Precision 0

Scale 0

Identity No

Identity Seed

Identity Increment

Is Rowid No

Formula

Collation



In the *extras* table *vid\_no* is the foreign key which refers to *info\_vehicle* table. This table stores the predefined extra details of vehicle for user to select some of them. The details are shown in the following table with examples.

Column Name	Refers	Example
vid_no	Vehicle id no	2
[Alloy wheels]	Alloy wheels	0
⋮	⋮	0
⋮	⋮	1
⋮	⋮	0

#### 10) *admin* table

Column Name	Data Type	Length	Allow Nulls
ad_id	int	4	
ad_name	varchar	50	
ad_pass	varchar	50	<input type="checkbox"/>

Columns

Description

Default Value

Precision

Scale

Identity

Identity Seed

Identity Increment

Is RowGuid

Formula

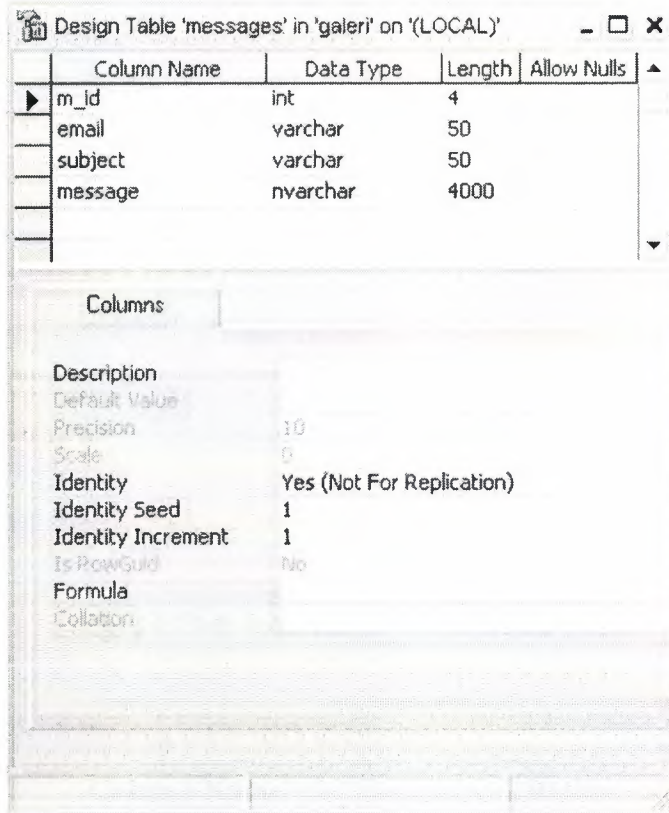
Collation

<database default>

In the *admin* table *ad\_id* is the primary key. This table stores the administrator's information. The details are shown in the following table with examples.

Column Name	Refers	Example
ad_id	Id no of the administrator	3
ad_name	Administrator's user name	Admin
ad_pass	Administrator's password	a1

### 11) messages table



In the *messages* table *m\_id* is the primary key. This table stores the messages which sent by the users to administrators. The details are shown in the following table with examples.

Column Name	Refers	Example
m_id	Message id no	157
email	Sender E-mail address	user@cmc.com
subject	Subject of the message	Help!!!
message	Received message	I couldn't find my car model

## 12) *links* table

Design Table 'links' in 'galeri' on '(LOCAL)'

Column Name	Data Type	Length	Allow Nulls
lid_no	int	4	
link	varchar	50	

Columns

Description

Link id no

Precision

10

Scale

0

Identity

Yes (Not For Replication)

Identity Seed

1

Identity Increment

1

Is RowGuid

No

Formula

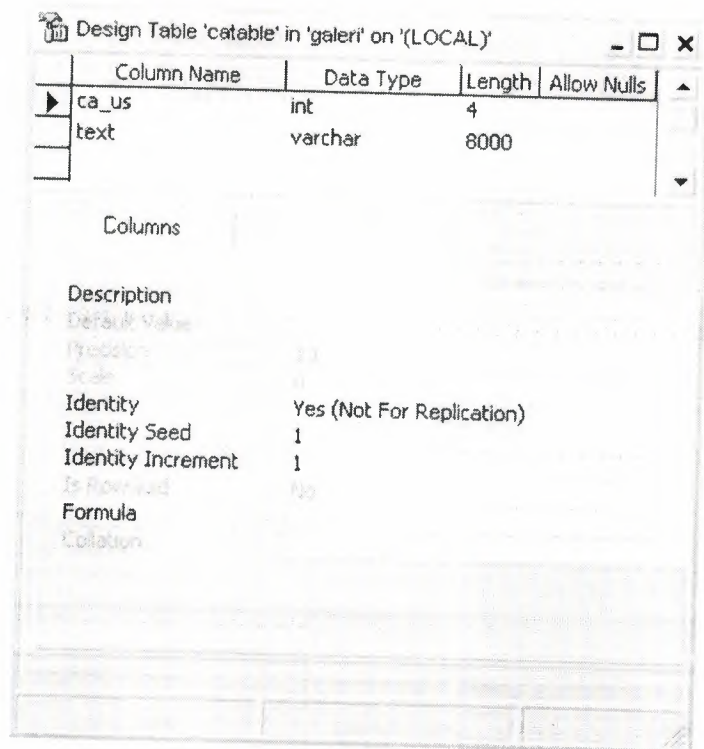
Collation

In the *links* table *lid\_no* is the primary key. This table stores the links which added by the administrators and it automatically added to the link page. The details are shown in the following table with examples.

Column Name	Refers	Example
lid_no	Link id no	87
link	Link address	<a href="http://www.neu.edu.tr">http://www.neu.edu.tr</a>



### 13) *catable* table



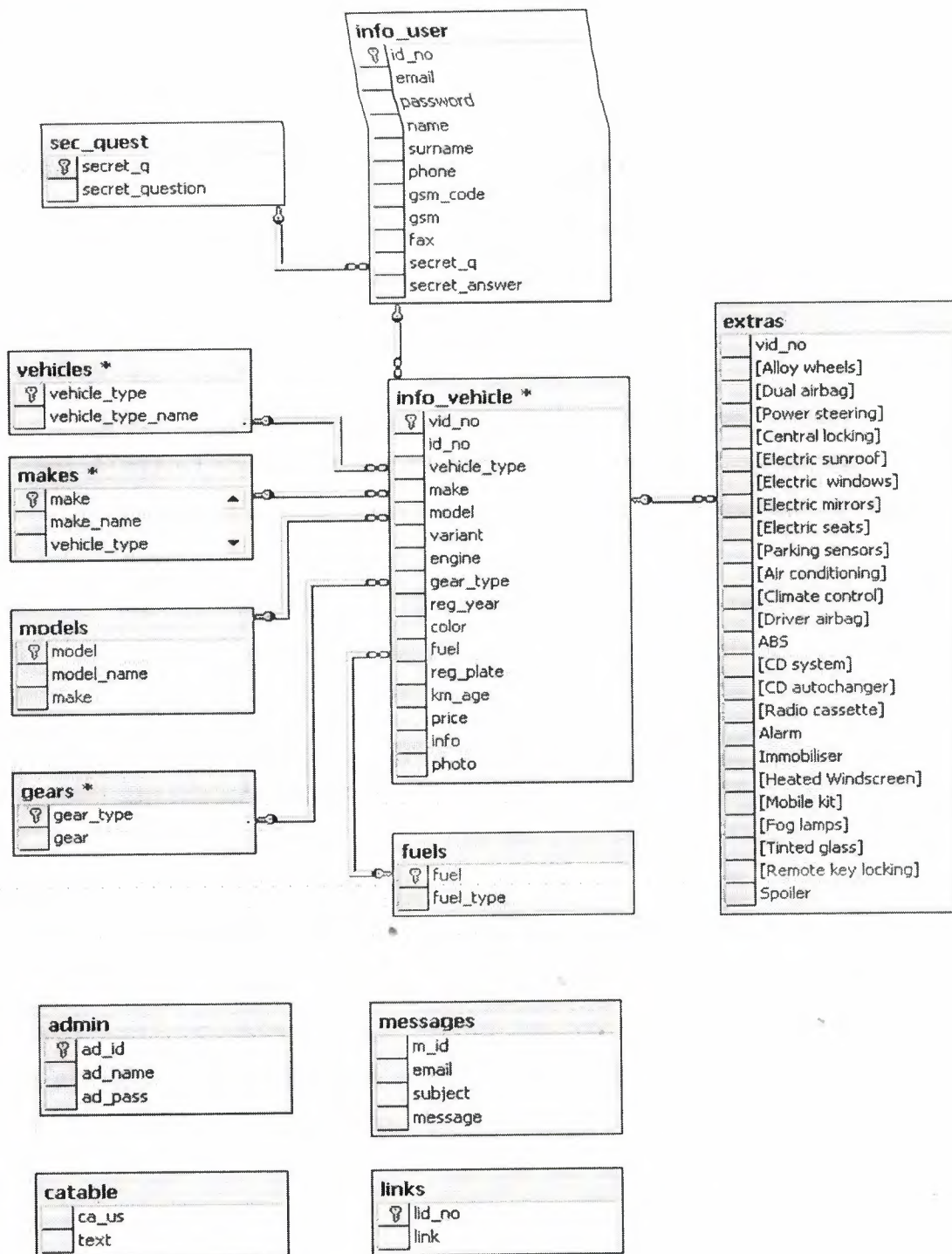
In the *catable* table *ca\_us* is keys of the contact us and about us contents. This table stores the contact us and about us pages contents which are added by the administrators and it automatically added to contact us and about us pages. The details are shown in the following table with examples.

Column Name	Refers	Example
ca_us	Defines the contact us or about us numbers	1
text	Contents texts	Tel: 2280000

With these database tables we can easily store the administrators, web page contents and web page members data.

### 5.3.2 Table Relations:

All the tables and their relations to each other are shown in the diagram below. From this relation diagram, the interaction points between the tables can be seen.



### 5.3.3 Database Environment:

For database purposes, MS SQL Server 2000 environment is used.

#### 5.3.3.1 Database Access:

Access to the database from the application is made by using ADO.NET. ADO.NET is the primary relational data access model for Microsoft .NET-based applications. It may be used to access data sources for which there is a specific .NET Provider, or, via a .NET Bridge Provider, for which there is a specific OLE DB Provider, ODBC Driver, or JDBC Driver.

ADO.NET uses some ADO objects, such as the Connection and Command objects. ADO.NET objects include the DataSet, DataReader, and DataAdapter.

**Connections:** For connection to and managing transactions against a database.

**Commands:** For issuing SQL commands against a database.

**DataReaders:** For reading a forward-only stream of data records from a SQL Server data source.

**DataSets:** For storing, remoting and programming against flat data, XML data and relational data.

**DataAdapters:** For pushing data into a DataSet, and reconciling data against a database.

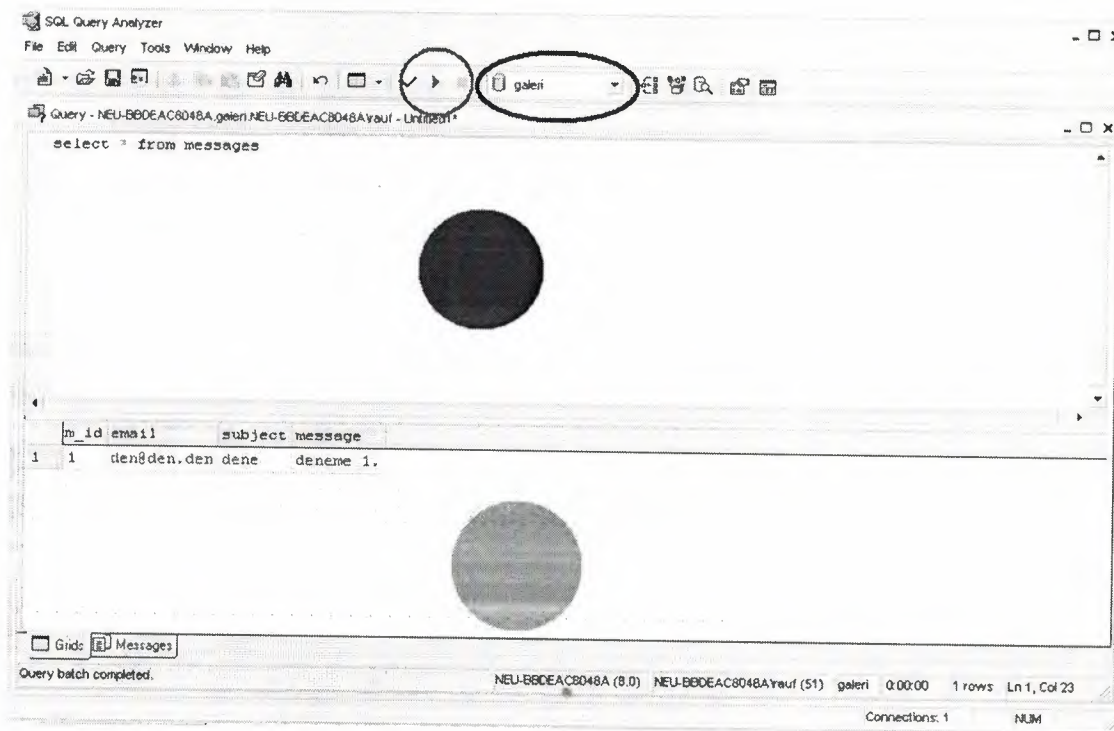
By the help of these objects access to the database is achieved. Sql query sentences are passed and executed with these objects. Also results of the queries are handled with these objects.

For example; to select data from a table simply "select \* from <table name> where <condition>" Query sentences is passed to the ADO.NET objects



### 5.3.4 SQL Query Analyzer

Finally, the SQL Query Analyzer provides us to check our SQL statements. While during web the program, it can be broken for missing or wrong SQL statement and it takes your time to finding and correcting wrong SQL statements. With SQL Query Analyzer you can check your statements before writing the program it is also the right way because of it keeps you away from the SQL statements errors in the program. Next figure shows the SQL Query Analyzer.



For using SQL Query Analyzer we have to know four main things about the SQL Query Analyzer page.

First of all, we have to choose the database in the combo box where the blue circle shows in. Secondly, we have to write SQL statement in the correct area where the blue point focuses on. Then we run the program with clicking the “Execute Query” button shown in red circle. Finally we get our results with the related rows in the grid

area where the orange point shows us, also with clicking messages you can learn how many rows selected.

## 5.4 Web Page Design (Code Side)

The most important side of the web page design is exactly coding side. In the coding side you have to be very careful be if you made a mistake while the page codes your project could not work properly. In this part of the project is about analyzing the page codes in three parts which are HTML, ASP.NET and C#. Let start the analyzing properly.

### 5.4.1 HTML Code Analyzing

Every HTML codes in this project starts with the same code because just it takes a little part of this project. But it is very important.

In the project HTML codes layouts as shown below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
  <HEAD>
    <title>Index</title>
    <meta name="vs_snapToGrid" content="False">
    <meta name="vs_showGrid" content="True">
    <meta name="GENERATOR" Content="Microsoft Visual Studio
.NET 7.1">
    <meta name="CODE_LANGUAGE" Content="C#">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </HEAD>
  <body ms_positioning="GridLayout" bgProperties="fixed"
background="file:///C:\Inetpub\wwwroot\ProjectFile\backgroundname.
JPG">
    <form id="FormName" method="post" runat="server">
    //////////////////////////////////////
```

```
</form></body></HTML>
```

Now it will explained these codes step by step:

An HTML document, written to the specifications of the World Wide Web Consortium (W3C), should begin with an element called the Document Type Definition, or DTD. Currently, the HTML 4.0 Transitional specification is in widespread use. A file written to that specification should have as its first line:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

This is a statement to the effect that this HTML document is on a publicly accessible server, obeying the W3C's conventions for HTML 4.0 Transitional. EN indicates the file is in English.

On the second line we have to write "html" tag to start programming in HTML

```
<HTML>
```

The head element can contain information about the document. The browser does not display the "head information" to the user.

On the third line the "head" tag

```
<HEAD>
```

The head element can contain information about the document. The browser does not display the "head information" to the user. The following tags can be in the head section: <base>, <link>, <meta>, <script>, <style>, and <title>.

On the fourth line its optional "title" tag is used as:

```
<title>
```

This element defines the title of the document.

From fifth to tenth line "meta" tags is used as:

```
<meta name="metaName" content="True/False">
```

The "meta" elements provide meta-information about your page, such as descriptions and keywords for search engines and refresh rates.

On the line thirteen the "body" tags is used as:



```
<body    ms_positioning="GridLayout"    bgProperties="fixed"
background="file:///C:\Inetpub\wwwroot\ProjectFile\backroun
dname.JPG">
```

The body element defines the documents' body. It contains all the contents of the document (like text, images, colors, graphics, etc.).

On the line sixteen "form" tags is used as:

```
<form id="FormName" method="post" runat="server">
```

The form element creates a form for user input. A form can contain textfields, checkboxes, radio-buttons and more. Forms are used to pass user-data to a specified URL.

At the last line the opened tags are closed orderly as:

```
</form></body></HTML>
```

As you see it is easy to working with HTML tags.

### 5.4.2 ASP.NET Code Analyzing

ASP.NET language keeps very important part of the project. While working with ASP.NET you have to understand this language very well and always working carefully. In the project C# is also used in code behind so to connect these two languages together you have write your connection string in ASP.NET side and also you have to well arrange both coding design. That means you have to know when you have to use ASP.NET and C# codes. After designing these codes you have to well coordinate the codes which are have to be work together.

Let start how the ASP.NET codes designed with code examples:

For integrating the ASP.NET with C# is used following code at the first line on the HTML code:

```
<%@ Page language="c#" Codebehind="webformname.aspx.cs"
AutoEventWireup="false" Inherits="Gra_Pro.WebForm15" %>
```

As you see in the example language is c# code behind is our web form name inherits is project name with web form name in the `<%@.....%>` statement.

For attaching the project logo the following code is used:

```
<asp:image id="Image1" style="Z-INDEX: 100; LEFT: 0px;
POSITION: absolute; TOP: 25px" runat="server"
ImageUrl="file:///C:\Inetpub\wwwroot\Gra-Pro\CMC_Logo-
goldton.JPG" Height="200px" Width="865px"></asp:image>
```

For adding the image buttons which are at the top of page use following code:

```
<asp:ImageButton id="about" style="Z-INDEX: 107; LEFT:
519px; POSITION: absolute; TOP: 0px" runat="server"
Height="25px" Width="173px"
ImageUrl="file:///C:\Inetpub\wwwroot\Gra-Pro\About.JPG"
TabIndex="40"></asp:ImageButton>
```

For constituting the links the next code written:

```
<asp:HyperLink id="sell" style="Z-INDEX: 104; LEFT: 125px;
POSITION: absolute; TOP: 366px" runat="server" Width="615px"
Height="40px" Font-Bold="True" Font-Italic="True" Font-
Names="Comic Sans MS" Font-Size="Large" BackColor="Yellow"
BorderColor="#8080FF" BorderStyle="Outset"
NavigateUrl="http://localhost/Gra-Pro/sign-in.aspx"
TabIndex="90">IF YOU WANT TO SELL YOUR CAR CLICK
HERE!!!</asp:HyperLink>
```

Now the following code is the label example which is used in the home page:

```
<asp:Label id="accmenu" style="Z-INDEX: 110; LEFT: 688px;
POSITION: absolute; TOP: 432px" runat="server">
```

```
Height="25px" Width="175px" Font-Size="Large" Font-
Names="Book Antiqua" Font-Bold="True"
ForeColor="Red">Account Menu:</asp:Label>
```

Let's show how the text box is generated on the site with an example:

```
<asp:TextBox id="emailtxt" style="Z-INDEX: 116; LEFT:
152px; POSITION: absolute; TOP: 320px" tabIndex="10"
runat="server" Width="200px" Height="20px"></asp:TextBox>
```

Hint: if you want to create password text box you have to add following string in the text box code:

```
TextMode="Password"
```

Hint: if you want to change your text box into multi line text box you have to put the code shown below:

```
TextMode="MultiLine"
```

For creating a drop down list you should write the following statement:

```
<asp:dropdownlist id="model" style="Z-INDEX: 108; LEFT:
8px; POSITION: absolute; TOP: 384px" runat="server"
Width="168px" Height="24px"
tabIndex="20"></asp:dropdownlist>
```

If you want to add radio button list the next statement which you should add your form:

```
<asp:RadioButtonList id="RadioButtonList1" style="Z-INDEX:
106; LEFT: 352px; POSITION: absolute; TOP: 232px"
runat="server" Width="260px" Height="8px" Font-Bold="True"
```



```

RepeatDirection="Horizontal"      ForeColor="Blue"      Font-
Size="Medium" tabIndex="1">
    <asp:ListItem                                Value="1"
Selected="True">CAR</asp:ListItem>
    <asp:ListItem Value="2">MOTORBIKE</asp:ListItem>
    <asp:ListItem Value="3">COMMERCIAL</asp:ListItem>
</asp:RadioButtonList>

```

In the form the following statement is written for creating the check box group:

```

<asp:CheckBoxList id="CheckBoxList1" style="Z-INDEX: 112;
LEFT: 8px; POSITION: absolute; TOP: 504px" runat="server"
Width="865px" Height="88px" RepeatColumns="8"
RepeatDirection="Horizontal" tabIndex="140">
    <asp:ListItem Value="0">Alloy wheels</asp:ListItem>
    <asp:ListItem Value="2">Dual airbag</asp:ListItem>
    <asp:ListItem Value="2">Power steering</asp:ListItem>
    <asp:ListItem                                Value="3">Central
locking</asp:ListItem>
    <asp:ListItem                                Value="4">Electric
sunroof</asp:ListItem>
    <asp:ListItem
Value="5">ElectricWindows</asp:ListItem>
    <asp:ListItem                                Value="6">Electric
mirrors</asp:ListItem>
    <asp:ListItem Value="7">Electric seats</asp:ListItem>
    <asp:ListItem                                Value="8">Parking
sensors</asp:ListItem>
    <asp:ListItem                                Value="9">Air
conditioning</asp:ListItem>
    <asp:ListItem                                Value="10">Climate
control</asp:ListItem>
    <asp:ListItem Value="11">Driver airbag</asp:ListItem>
    <asp:ListItem Value="12">ABS</asp:ListItem>
    <asp:ListItem Value="13">CD system</asp:ListItem>
    <asp:ListItem Value="14">CDautochanger</asp:ListItem>

```

```

<asp:ListItem                                Value="15">Radio
cassette</asp:ListItem>
<asp:ListItem Value="16">Alarm</asp:ListItem>
<asp:ListItem Value="17">Immobiliser</asp:ListItem>
<asp:ListItem                                Value="18">Heated
windscreen</asp:ListItem>
<asp:ListItem Value="19">Mobile kit</asp:ListItem>
<asp:ListItem Value="20">Fog lamps</asp:ListItem>
<asp:ListItem Value="21">Tinted glass</asp:ListItem>
<asp:ListItem                                Value="22">Remote          key
locking</asp:ListItem>
<asp:ListItem Value="23">Spoiler</asp:ListItem>
</asp:CheckBoxList>

```

For the showing the result the data grid is used as the next code:

```

<asp:DataGrid id="DataGrid1" style="Z-INDEX: 112;
LEFT: 88px; POSITION: absolute; TOP: 360px"
runat="server" Height="192px" Width="720px"
BorderColor="#E7E7FF" BackColor="White"
BorderStyle="None" BorderWidth="1px" CellPadding="3"
GridLines="Horizontal" AllowPaging="True"
TabIndex="1">
    <SelectedItemStyle Font-Bold="True"
    ForeColor="#F7F7F7"
    BackColor="#738A9C"></SelectedItemStyle>
    <AlternatingItemStyle
    BackColor="#F7F7F7"></AlternatingItemStyle>
    <ItemStyle ForeColor="#4A3C8C"
    BackColor="#E7E7FF"></ItemStyle>
    <HeaderStyle Font-Bold="True"
    ForeColor="#F7F7F7"
    BackColor="#4A3C8C"></HeaderStyle>
    <FooterStyle ForeColor="#4A3C8C"
    BackColor="#B5C7DE"></FooterStyle>

```

```

<PagerStyle
    HorizontalAlign="Right"
    ForeColor="#4A3C8C"
    BackColor="#E7E7FF"
    Mode="NumericPages"></PagerStyle>
</asp:DataGrid>

```

There is also some error control coding which are not caused by codes, caused by users, in ASP.NET called validation now it will shown some examples about validation controls. First code is about compare validation which compares two inputs:

```

<asp:CompareValidator id="CompareValidator1" style="Z-
INDEX: 134; LEFT: 336px; POSITION: absolute; TOP: 352px"
runat="server" Width="160px" Height="24px"
ErrorMessage="does not match"
ControlToCompare="passwordtxt"ControlToValidate="conftxt">
</asp:CompareValidator>

```

The other validation type is required filled validator which does not allow to process without an input:

```

<asp:RequiredFieldValidator id="RequiredFieldValidator1"
style="Z-INDEX: 145; LEFT: 304px; POSITION: absolute; TOP:
304px" runat="server" Width="16px" Height="16px"
ErrorMessage="*"
ControlToValidate="emailtxt"></asp:RequiredFieldValidator>

```

Another validation control is regular expression validator which constraints the input:

```

<asp:RegularExpressionValidator
id="RegularExpressionValidator3" style="Z-INDEX: 142; LEFT:
568px; POSITION: absolute; TOP: 456px"
runat="server" Width="208px" Height="24px"
ControlToValidate="gsmno" ErrorMessage="Don 't use Space or
Characters" ValidationExpression="\d{7}">
</asp:RegularExpressionValidator>

```



The last type of validation of which is used in the project is custom validator shown following example:

```
<asp:customvalidator id="incorr" style="Z-INDEX: 115; LEFT: 264px; POSITION: absolute; TOP: 352px" runat="server" Height="32px" Width="184px" Display="Dynamic">Incorrect E-mail or Password </asp:customvalidator>
```

### 5.4.3 C# Code Analyzing

The most important side of the project is C# coding because of the most usable language you can do everything with C# codes. Also the most advantage thing is, C# is an Object Oriented Programming Language. And the best side in C# is in perfect harmony with .NET Technology. In this project it the very big advantages of C# and ASP.NET are work in good adaptation. C# is used in the project to do several things. Now some of them are shown. The examples which are written below are in the project real codes.

In code behind we have to define "using" and namespaces states as shown below:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
namespace ProjectName
```

In the above example is used "using System.Data.SqlClient;" only the pages where the database is connected.

After these statements we have to define the class which we will use in the project. The following codes for an example:

```
public class WebForm3 : System.Web.UI.Page
{
    protected System.Web.UI.WebControls.Label Email;
    protected System.Web.UI.WebControls.TextBox emailtxt;
    protected System.Web.UI.WebControls.HyperLink click;
    protected System.Web.UI.WebControls.Image Image1;
    protected System.Web.UI.WebControls.Button Button1;
    protected System.Web.UI.WebControls.ImageButton contact;
    protected System.Web.UI.WebControls.RadioButtonList Radiol;
    protected System.Web.UI.WebControls.DropDownList make;
    protected System.Web.UI.WebControls.CheckBoxList ChBxList1;
    protected System.Web.UI.WebControls.DataGrid DataGrid1;
    protected System.Web.UI.WebControls.CompareValidator CompV;
    protected System.Web.UI.WebControls.CustomValidator incorr;
    protected System.Web.UI.WebControls.RequiredFieldValidator;
    protected System.Web.UI.WebControls.RegularExpresinValator;
    *
    //////////////////////////////////////
}
```

\*Note: when the programmers want to connect the page to the database they write connection string on the "\*" line as:

```
SqlConnection con = new SqlConnection("Database=galeri;
Server=NEU-BBDEAC8048A; Integrated Security=SSPI;");
```

After writing these statements they would start to write their simple codes for simple buttons such as forwarding a page which is shown below:

```
private void home_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
{
    Response.Redirect("index.aspx");
}
```

```
}
```

In sign in page the E-mail and password are checked with connecting the database. As an example:

```
string sql;
sql=" SELECT * " +
" FROM info_user " + " WHERE email = '" +
emailtxt.Text.ToString()+ "' AND password = '" +
passwordtxt.Text.ToString() + "'";

con.Open();
SqlDataAdapter da = new SqlDataAdapter(sql,con);
DataSet ds = new DataSet();
da.Fill(ds);

if (ds.Tables[0].Rows.Count>0)
    { Response.Redirect("useropp.aspx");
    }
else
    { Response.Redirect("index.aspx");
    }
```

Then there is some functions to check user operation with developing previous code:

```
private bool checkuser()
{
    try
    {
        string sql;
        sql=" SELECT * " +
        " FROM info_user " +
        " WHERE email = '" + emailtxt.Text.ToString()+ "' AND
        password = '" + passwordtxt.Text.ToString() + "'";
        con.Open();
        SqlDataAdapter da = new SqlDataAdapter(sql,con);
        DataSet ds = new DataSet();
        da.Fill(ds);

        if (ds.Tables[0].Rows.Count>0)
        {
            string sec;
            sec=ds.Tables[0].Rows[0]["secret_q"].ToString();

            con.Close();
            return true;
        }
    }
}
```



```

else
{
    string sql3;
    sql3=" SELECT * " +
    " FROM info_user " +
    " WHERE email = '" + emailtxt.Text.ToString()+ "' " ;
    SqlDataAdapter da3 = new SqlDataAdapter(sql3,con);
    DataSet ds3 = new DataSet();
    da3.Fill(ds3);
    string sec;
    sec=ds3.Tables[0].Rows[0]["secret_q"].ToString();
    string sql2;
    sql2=" SELECT * " +
    "FROM sec_quest " +
    "WHERE secret_q = '" + sec + "'";
    SqlDataAdapter da2= new SqlDataAdapter(sql2,con);
    DataSet ds2= new DataSet();
    da2.Fill(ds2);

    Session["email"]=ds3.Tables[0].Rows[0]["email"].ToString();
    Session["secret_question"]=ds2.Tables[0].Rows[0]["secret_question"].ToString();
    con.Close();
    return false;
}
catch(Exception e)
{
    return false;
}
}

```

```

private void Button1_Click(object sender, System.EventArgs e)
{
    if(checkuser())
        Response.Redirect("sign-in.aspx");
    else
        Response.Redirect("forget.aspx");
}

```

In the previous code some codes are explained on the next part:

In the C# sharp it's easy to write your SQL statements in defined string as in this code. After defining SQL statements you should open your database as `con.Open()`; then you should create a data adapter like `SqlDataAdapter da= new SqlDataAdapter();` in the parenthesis you have to write your SQL string with your connection name such as `con` and then you should define a data set for example `DataSet ds= new DataSet();`

And you can fill your data Adapter with the data which are stored in your data set such as `da.Fill(da);` in this example there is some sessions which are using for carrying the data between the pages. You can also use sessions as `Session["name"]="Jones";`.

As you in the example it check the database in a Boolean function and then it check the data when the user clicks the button then it forward user to a page which depends on the answer of Boolean function.

Let's show some examples about the SQL usages in C#.

First how to fill a drop down list from the database:

```
string sql = "SELECT make_name FROM makes";
SqlDataAdapter damake = new SqlDataAdapter (sql,con);
DataSet dsmake = new DataSet();
damake.Fill(dsmake, "makename");

make.DataSource = dsmake.Tables["make_name"];
make.DataValueField = "make_name";
make.DataTextField="make";
make.DataBind();
make.Items.Insert(0, "");
```

Now it will show how to fill the data grid from the database:

```
SqlDataAdapter da= new SqlDataAdapter(sql,con);
DataSet ds= new DataSet();
da.Fill(ds,"info_vehicle");
vehiclegrid.DataSource= ds;
vehiclegrid.DataBind();
con.Close();
```

After theses operations next code shows how data delete from the database:

```
sql=" DELETE FROM      info_user, info_vehicle,  WHERE
id_no='" + idtxt.Text.ToString() + "' ";
con.Open();
SqlCommand cmd = new SqlCommand(sql,con);
cmd.ExecuteNonQuery();
con.Close();
```

## CONCLUSION

In the past, today and the future, the reason of the developing the technology is to make human life easier. Today computer sciences are the fastest grooving technology of the world to make life easier. Because of the internet, people do their works in short time and spending less effort and also more economically. In the growing internet technologies one of the important sides is E-commerce. As you see, this project is about interactive web page design for car gallery which is a kind of E-commerce.

Today people prefer to see all the possibilities, all the alternatives in one environment in order to make a comparison while shopping. This is also valid in vehicle shopping case. Internet is the best way to manage this. In order to go to a car bazaar or go to lots of car galleries one by one, all the information you need is all on a web site. This lets you to save your time, money and effort. All the alternatives, all the detail information and much more is just near as a mouse click.

After the business side of this project, technical issues of the project can be mentioned. This project obtains us not only about interactive web page design for car gallery but also about HTML, ASP.NET, .NET Technologies, C#, MS SQL Server 2000 and also their fundamentals, structures and how we should use these technologies, programming languages, database systems, and how we generate our codes and database diagrams while designing a web site.

The E-commerce web pages such as car galleries, which is the subject of this project, can be improved with some other branches. Configuring cars, buying vehicles directly from the web or even making a public auction to sell vehicles, after ordering a car following the steps and paths your car passes through till it arrives to you, making the insurance operations via these types of web sites are some of those new development branches. These new improvement subjects can be increased.

E-commerce is the rising star in the new business life. It is the most preferred one because of its simplicity and wide scope. It seems like, days that all commercial operations will be done on the web, is not so far.



## REFERENCES

- [1] Castro Elizabeth, *HTML for the World Wide Web with XHTML and CSS: Visual QuickStart Guide, Student Edition*, Peachpit Press
- [2] Ian Lloyd, *Build Your Own Web Site The Right Way Using HTML & CSS*, published by SitePoint.
- [3] Adrian Turttschi, Jason Werry, Greg Hack, Joseph Albahari, *C#.NET Web Developer's Guide*, Syngress Publishing, Inc.
- [4] Fritz Onion, *Essential ASP.NET with Examples in C#*, Addison Wesley press
- [5] ZakRuvalcaba, *Build Your Own ASP.NET Website Using C# & VB.NET*, published by SitePoint.
- [6] Robert Vieira, *Professional SQL Server 2000 Programming*, WROX press
- [7] Antony T.Mann *Microsoft SQL Server 2000 For Dummies* Wiley
- [8] <http://www.htmlgoodies.com>
- [9] <http://www.w3schools.com>
- [10] <http://www.dotnettv.com>