NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

WEB APPLICATIONS OF FAST - FOOD SALE AND RESTAURANT SALE MANAGEMENT

Graduation Project
COM – 400

Student:	Engin Alan
Supervisor:	Ümit Soyer

Nicosia - 2006

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Mr. Ümit Soyer for his invaluable advice and confidence to my working style. If my project is not same with the one I determined four months ago, but more developed, well-thought and constructed, it is cause of him.

Second, I would like to thank Assoc. Pr. Dr. Rahib Abiyev for teaching programming methods in C languages, so it has been possible to work on the project via a widely used programming language.

Third, I would like to thank my best friend Mr. Önder Tellioğlu (Microsoft CRM Software Professional) for his valuable advices and patience to my incessant questions.

Fourth, I would like to my family, especially to my mother for every day calling and motivation.

Finally, I would also like to thank my homemates Göktuğ Ataç and Mesut Arık, also my friends Süleyman Kerimoğlu, Sinan Özerenler, Ozan Akkoca and H. Rauf Dilsiz for their advice, support and enjoyable jokes to entertain me when I was very anxious.

ABSTRACT

Web applications are program structures which is working on the internet and communicating with servers. They consist of two sections as client side interface and server side database application.

Presented project is a web application which has two sections.

First section is a web page which is used to sale fast-food products on the internet and support an advertisement interface for customer. In this section, customers are provided to submit their information to have a membership for shopping and having the products by using this membership. An ease to choose a popular product group is presented to customers so, they have been able to shorten the shopping time. All orders are hold on a server.

Second section is a web interfaced program for restaurants to enter the system by using a special login name and password. Restaurants manage orders via this program.

In the project, code examples are shown from application. SQL Server 2000, HTML 4.0 and VISUAL STUDIO .NET 2000 platform are included in the project with their major attributes. ASP.NET is presented both in .NET platform and project structer sections.

TABLE OF CONTENTS

INTRODUCTION

It is very difficult to serve people for their requirements nowadays. Food industry is one of the way to provide a service. It has been increasing the number of companies supporting products to meet the need of food on the world.

While the companies do their sale from their member shops and markets directly, an alternative way known as e-commerce has been having importance explicitly. Besides the existence of the internet, food industry has needed to change their sale strategy and leave one side serving policy.

As an objective of providing companies support on internet sale and management tools for their sale terminals, this project is prepared as two sections. Both sections are web application programmed on .NET platform with ASP.NET and C# code behind. Working on the server and new web technologies of present are the main reasons to prepare this project as web application.

Project has been presented as four chapter:

Chapter One presents the existence of the web and internet shortly and includes properties of HTML, the most famous markup language for web pages, and its tags. Also, similarities and differences of HTML and XHTML are included.

Chapter Two presents database management system from the days in whics files were used to hold data up to now that modern systems have been used shortly. This chapter includes the properties of the SQL Server 2000 Database and the way to use it, mostly. The program of database system and query language is differentiated in the chapter and took up in different sections.

Chapter Three demonstrate Visual Studio .NET 2003 platform with general structure and features of ASP.NET. Files that constituting a web application and their attributes are denoted in this chapter.

Chapter Four presents the working principles of web applications of the project. Requirements of the system and most used and important forms of applications are investigated by using user interface parts of project.

Chapter Five presents the characteristics of the project and demonstrates the code structure. Object oriented programming structure and predefined system classes and functions are included via examples from the application.

CHAPTER ONE: WORLD WIDE WEB AND HTML – XHTML

1.1 The Internet

Though it began as a military experiment and spent its adolescence as a sandbox for academics and eccentrics, in less than a decade the worldwide network of computer networks - also known as the Internet - has matured into a highly diversified, financially important community of computer users and information vendors.

In many ways, the Web - the open community of hypertext-enabled document servers and readers on the Internet - is responsible for the meteoric rise in the network's popularity. You, too, can become a valued member by contributing: writing HTML and XHTML documents and then making them available to web surfers worldwide.

Although popular media accounts are often confused and confusing, the concept of the Internet really is rather simple: it's a worldwide collection of computer networks - a network of networks - sharing digital information via a common set of networking and software protocols.

Networks are not new to computers. What makes the Internet unique is its worldwide collection of digital telecommunication links that share a common set of computer-network technologies, protocols, and applications. Whether you run Microsoft Windows XP, Linux, Mac OS X, or even the now ancient Windows 3.1, when connected to the Internet, computers all speak the same networking language and use functionally identical programs, so you can exchange information - even multimedia pictures and sound - with someone next door or across the planet.

1.2 Html And The Web

It tooks another spark to light the Internet rocket. At about the same time the Internet opened up for business, some physicists at CERN, the European Particle Physics Laboratory, released an authoring language and distribution system they developed for creating and sharing multimedia-enabled, integrated electronic documents over the Internet. And so was born Hypertext Markup Language (HTML), browser software, and the Web. No longer did authors have to distribute their work as fragmented collections of pictures, sounds, and text. HTML unified those elements.

Moreover, the Web's systems enabled hypertext linking, whereby documents automatically reference other documents located anywhere around the world: less rummaging, more productive time online.

1.3 Clients, Servers, And Browsers

The Internet connects two kinds of computers: servers, which serve up documents, and clients, which retrieve and display documents for us humans. Things that happen on the server machine are said to be on the server side, while activities on the client machine occur on the client side.

To access and display HTML documents, we run programs called browsers on our client computers. These browser clients talk to special web servers over the Internet to access and retrieve electronic documents.

Several web browsers are available (most for free), each offering a different set of features. For example, browsers like Lynx run on character-based clients and display documents only as text. Others run on clients with graphical displays and render documents using proportional fonts and color graphics on a 1024 x 768, 24-bit-per-pixel display. Others still - Netscape Navigator, Microsoft's Internet Explorer, and Opera, to name the leading few - have special features that allow you to retrieve and display a variety of electronic documents over the Internet, including audio and video multimedia.

All web activity begins on the client side, when a user starts his or her browser. The browser begins by loading a home page document, either from local storage or from a server over some network, such as the Internet, a corporate intranet, or a town extranet. In these latter cases, the client browser first consults a domain name system (DNS) server to translate the home page document server's name, such as www.oreilly.com, into an IP address, before sending a request to that server over the Internet. This request (and the server's reply) is formatted according to the dictates of the Hypertext Transfer Protocol (HTTP) standard.

A server spends most of its time listening to the network, waiting for document requests with the server's unique address stamped on them. Upon receipt of a request, the server verifies that the requesting browser is allowed to retrieve documents from the server and, if so, checks for the requested document. If found, the server sends (downloads) the document to the browser. The server usually logs the request, the client computer's name, the document requested, and the time.

Back on the browser, the document arrives. If it's a plain-vanilla ASCII text file, most browsers display it in a common, plain-vanilla way. Document directories, too, are treated like plain documents, although most graphical browsers display folder icons that the user can select with the mouse to download the contents of subdirectories.

Browsers also retrieve context files from a server. Unless assisted by a helper program or specially enabled by plug-in software or applets, which display an image or video file or play an audio file, the browser usually stores downloaded binary files directly on a local disk for later use.

For the most part, however, the browser retrieves a special document that appears to be a plain text file but that contains both text and special markup codes called tags. The browser processes these HTML or XHTML documents, formatting the text based on the tags and downloading special accessory files, such as images.

The user reads the document, selects a hyperlink to another document, and the entire process starts over.

1.4 HTML And XHTML

1.4.1 What They Are

HTML and XHTML are document-layout and hyperlink-specification languages. They define the syntax and placement of special, embedded directions that aren't displayed by the browser but tell it how to display the contents of the document, including text, images, and other support media. The languages also tell you how to make a document interactive through special hypertext links, which connect your document with other documents - on either your computer or someone else's - as well as with other Internet resources.

1.4.2 What They Aren't

Despite all their new, multimedia-enabling page-layout features, and the hot technologies that give life to HTML/XHTML documents over the Internet, it is also important to understand the languages' limitations. They are not word-processing tools, desktop-publishing solutions, or even programming languages. Their fundamental purpose is to define the structure and appearance of documents and document families so that they may be delivered quickly and easily to a user over a network for rendering on a variety of display devices. Jack of all trades, but master of none, so to speak.

1.4.3 Content Versus Appearance

HTML and its progeny, XHTML, provide many different ways to let you define the appearance of your documents: font specifications, line breaks, and multicolumn text are all features of the language. Of course, appearance is important, since it can have either detrimental or beneficial effects on how users access and use the information in your documents.

Nonetheless, we believe that content is paramount; appearance is secondary, particularly since it is less predictable, given the variety of browser graphics and text-formatting capabilities. In fact, HTML and XHTML contain many ways for structuring your document content without regard to the final appearance: section headers, structured lists, paragraphs, rules, titles, and embedded images are all defined by the standard languages without regard for how these elements might be rendered by a browser. Consider, for example, a browser for the blind, wherein graphics on the page come with audio descriptions and alternative rules for navigation. The HTML/XHTML standards define such a thing: content over visual presentation.

If you treat HTML or XHTML as a document-generation tool, you will be sorely disappointed in your ability to format your document in a specific way. There is simply not enough capability built into the languages to allow you to create the kinds of documents you might whip up with tools like FrameMaker or Microsoft Word. Attempts to subvert the supplied structuring elements to achieve specific formatting tricks seldom work across all browsers.

In short, don't waste your time trying to force HTML and XHTML to do things they were never designed to do.

1.4.4 Structure of HTML Document

HTML and XHTML documents consist of text, which defines the content of the document, and tags, which define the structure and appearance of the document. The structure of an HTML document is simple, consisting of an outer <html> tag enclosing the document head and body.

Each document has a head and a body, delimited by the <head> and <body> tags. The head is where you give your document a title and where you indicate other parameters the browser may use when displaying the document. The body is where you put the actual contents of the document. This includes the text for display and document-control markers (tags) that advise the browser how to display the text. Tags also reference special-effects files, including graphics and sound, and indicate the hot spots (hyperlinks and anchors) that link your document to other documents.

For the most part, tags - the markup elements of HTML and XHTML - are simple to understand and use, since they are made up of common words, abbreviations, and notations. For instance, the <i> and </i> tags respectively tell the browser to start and stop italicizing the text characters that come between them. Accordingly, the syllable "simp" in our barebones example above would appear italicized on a browser display.

The HTML and XHTML standards and their various extensions define how and where you place tags within a document. Let's take a closer look at that syntactic sugar that holds together all documents.

Every tag consists of a tag name, sometimes followed by an optional list of tag attributes, all placed between opening and closing brackets (< and >). The simplest tag is nothing more than a name appropriately enclosed in brackets, such as <head> and <i>. More complicated tags contain one or more attributes, which specify or modify the behavior of the tag.

According to the HTML standard, tag and attribute names are not case-sensitive. There's no difference in effect between <head>, <Head>, <HEAD>, or even <HeaD>; they are all equivalent. With XHTML, case is important: all current standard tag and attribute names are in lowercase. For both HTML and XHTML, the values that you assign to a particular attribute may be case-sensitive, depending on your browser and server. In particular, file location and name references - or uniform resource locators (URLs) - are case-sensitive.

Tag attributes, if any, belong after the tag name, each separated by one or more tab, space, or return characters. Their order of appearance is not important. A tag attribute's value, if any, follows an equals sign (=) after the attribute name. You may include spaces around the equals sign, so that width=6, width = 6, width =6, and width= 6 all mean the same. For readability, however, we prefer not to include spaces. That way, it's easier to pick out an attribute/value pair from a crowd of pairs in a lengthy tag.

With HTML, if an attribute's value is a single word or number (no spaces), you may simply add it after the equals sign. All other values should be enclosed in single or double quotation marks, especially those values that contain several words separated by spaces. With XHTML, all attribute values must be enclosed in double quotes. The length of the value is limited to 1,024 characters.

Comments are another type of textual content that appears in the source HTML document but is not rendered by the user's browser. Comments fall between the special <!-- and --> markup elements. Browsers ignore the text between the comment character sequences.

There must be a space after the initial <!-- and preceding the final -->, but otherwise you can put nearly anything inside the comment. The biggest exception to this rule is that the HTML standard doesn't let you nest comments.

1.4.5 What is CSS?

HTML or XHTML only divide a document up into paragraphs, lists, headings and so on, but does not really say how these things should look. Rather, a browser generally makes some assumptions about how things should look -- and we're then stuck with those choices.

This could be changed given a way of controlling how different markup elements (like headings, paragraphs, etc) look. This is the role of CSS. Cascading Style Sheets, or CSS, is a language, separate from HTML or XHTML, designed for specifying the layout or formatting properties of the various HTML elements in a document. For example, a CSS statment like the following

```
Body
{       font-family: Arial,helvetica,sans-serif;
        color: black;
        background-color: white;
}
```

means that, inside the BODY of a document, the desired font is Arial, the desired text color is black, and the desired background color for the page is white. More complicated rules let you control underlining of links, the placement of background images, the widths of margins, the colors of borders around paragraphs or headings, etc. As an example, the page you are viewing has an "attached" style sheet to control how it looks. This stylesheet is included into this document using a special link element of the form: <link rel="stylesheet" href="stylesheet.css" > where stylesheet.css contains the CSS document.

1.4.6 HTML Tags As Group Tables

Table 1.1: Structure Tags

Function	**Start Tag**	**Attributes**	**End Tag**
HTML File	<html>	None	</html>
File Header	<head>	None	</head>
File Title	<title>	None	</title>
Comments	<!--	Your comments go between the start and end tags. Put a space between the -- and your comments.	-->
Body	<body>	Background="filename"	</body>
		bgcolor="color value"	
		text="color value"	
		link="color value"	
		vlink="color value"	
Division	<div>	align="right/left/center"	</div>
		style="property:value;"	
		class="classname"	
Span (inline)	<span>	style="property:value;"	</span>
		class="classname"	

Figure 1.1: Explanation of Structure Tags

Table 1.2: Basic Text Tags

Function	Start Tag	Attributes	End Tag
Line Break	<br>	clear="left/right/all"	</br> or <br />
Paragraph	<p>	align="center/right"	</p>
Bold	<b>	none	</b>
Italic	<i>	none	</i>
Typewriter Text	<tt>	none	</tt>
Headline	<h1-6>	align="center/right"	</h1-6>
Font	<font>	face="name, name"	</font>
		size="+/-value/fixed size"	
		color="color value"	
		Note: the font tag is being phased out in favor of CSS styles.	
Horizontal Rule	<hr>	size="XX"	</hr> or <hr />
		width="XX/XX%"	
		noshade	
Block Quote	<blockquote>	none	</blockquote>

Figure 1.2: Explanation of Basic Text Tags

Table 1.3: Link Tags

Function	Start Tag	Attributes	End Tag
Anchor Link	<a>	href="filename"	</a>
		target="windowname"	
Anchor Mark	<a>	name="markname"	</a>

Figure 1.3: Explanation of Link Tags

Table 1.4: Image Text

Function	Start Tag	Attributes	End Tag
Insert Image	<img>	src="filename"	</img>
		align="left/right"	
		width="XXX"	
		height="XXX"	
		alt="text that desribes image"	
		ISMAP	
		USEMAP="#mapname"	

Figure 1.4: Explanation of Image Tags

Table 1.5: Form Tags

Function	**Start Tag**	**Attributes**	**End Tag**
Form	<form>	method=get/put	</form>
		action="programname"	
Input Field	<input>	name="variablename"	</input>
		type=text/password/	
		checkbox/radio/submit/	
		reset/image	
Selection List	<select>	name="variablename"	</select>
		size=XX	
		multiple	
Selection Option	<option>	none	</option>
Scrolling Text Field	<textarea>	name="variablename"	</textarea>
		rows=XX	

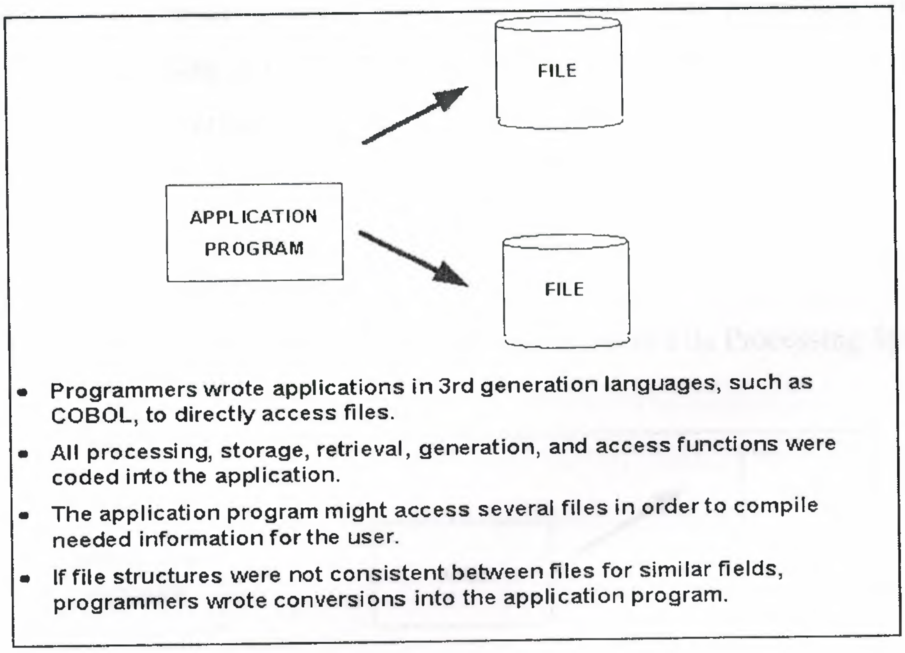
Figure 1.5: Explanation of Form Tags

CHAPTER TWO: DATABASE MANAGEMENT WITH SQL SERVER 2000 DEVELOPER EDITION

2.1 Evolution of Database Management System

2.1.1 File - Processing System:

In early processing systems, an organization's information was stored as groups of records in separate files. These file processing systems consisted of a few data files and many application programs. Each file, called a flat file, contained and processed information for one specific function, such as accounting or inventory. Programmers used programming languages such as COBOL to write applications that directly accessed flat files to perform data management services and provide information for users.



- Programmers wrote applications in 3rd generation languages, such as COBOL, to directly access files.
- All processing, storage, retrieval, generation, and access functions were coded into the application.
- The application program might access several files in order to compile needed information for the user.
- If file structures were not consistent between files for similar fields, programmers wrote conversions into the application program.

Figure 2.1: File Processing System

Although File processing system is a great improvement over manual record keeping systems, they suffer from these limitations:

Separated and Isolated Data: To make a decision, a user might need data from two separate files. First, the files were evaluated by analysts and programmers to determine the specific data required from each file and the relationships between the data. Then applications could be written in some language like COBOL to process and extract the needed data.

Data Duplication: Often, the same information was stored in more than one file. In addition to taking up more file space on the system, this replication of data caused loss of data integrity. For instance, if a customer's address were stored in four different files, an address change would have to be updated in each file separately. If a user was not consistent in updating all files, no one would know which information was correct.

Application Program Dependency: In File processing system application program depended on the file formats. In file processing system the physical formats of the files and records are part of the application code. So any change in the records require change of the application code and thus made things very complex.

Incompatible Files: File formats depend on the language of the product used to generate them. So if one file is processed using C and other using VB, they cannot be combined and processed. For getting information combined from them we need to convert both files to a common structure

Information Providing as per users perspectives: In file processing system it is difficult to represent data in Users perspective because it is difficult to readily join the information between various files. So it was difficult to answer ad-hoc queries of database users.

2.1.2 Database Management System:

DBMS able to take care of most of the limitations of File Processing System.

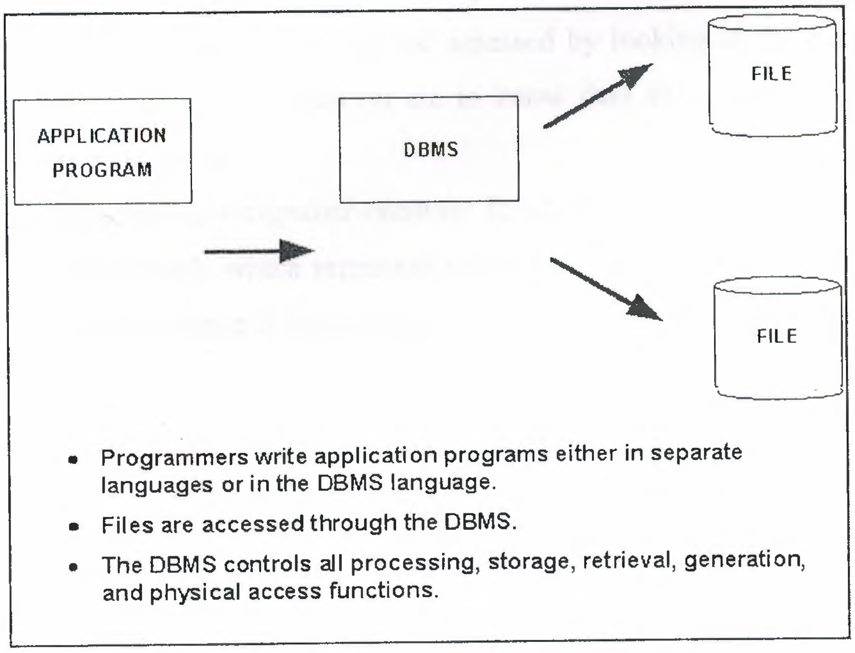

APPLICATION PROGRAM
DBMS
FILE
FILE
• Programmers write application programs either in separate languages or in the DBMS language.
• Files are accessed through the DBMS.
• The DBMS controls all processing, storage, retrieval, generation, and physical access functions.

Figure 2.2: Database Management System

Separated and Isolated Data issue: In DBMS all the data is stored in a single facility called the database. It's very easy and simple to combine the data in DBMS. Programmer just need to tell DBMS that how are the data to be combined and the DBMS performs the necessary action. Thus the Programmer does not need to write the program to consolidate the files.

Reduce Data Duplication: Since all the data is integrated the information need to be stored at just one place and unlike File Processing system it does not need to be kept at more then one place.

Program/ Data Dependence: Database reduces the dependency of programs on file formats. All record formats are stored in the database itself by the DBMS and not by application Programs so there is no need for database application program to include file format and records the process.

Allows for representation of data from the user's view: Relationships between data objects in a user's environment are stored in the DBMS. Data elements from any number of files can be combined to create useful forms, reports, and other applications.

2.2 Characteristics of Database And DBMS

2.2.1 Characteristics of Database

It is self-describing: In addition to the users source data, a data dictionary or metadata contain information about the database structure. In this sense database is similar to Library. This characteristic is important as it promoted program- data independence. So database structure can be assessed by looking at the database itself and we do not need any documentation etc to know data structure. Also changes to database are easy to be made.

It is a collection of integrated records: In addition to Meta data and user data database contains Indexes, which represent relationship between among the data and also improve the performance of database applications. Thus a database contains 4 types of data:

- Users Data
- Meta Data
- Indexes
- Application Metadata

A Database is Model of Model: Its representation of the users world or users model. Thus database is a model of users' model of their business activities.

Therefore based on the characteristics of the Database we can formulate the definition as: A database is self-describing collection of integrated records.

2.2.2 Characteristics of Database Management System

- It is collection of databases
- It is a computerized record-keeping system
- It contains facilities that allow the user to:
  - Add and delete files to the system
  - Insert, retrieve, update, and delete data in existing files
- A DBMS may contain many databases that can be used for separate purposes or combined to provide useful information

Therefore based on this we can define Database Management System as: A database management system (DBMS) is a program that allows users to define, manipulate, and process the data in a database in order to produce meaningful information.

Functions of a database management system can be ordered as:

- To organize data
- To store data
- To control access to data
- To protect data

Usage of database management systems can be ordered as:

To provide decision support: Managers and analysts retrieve information generated by the DBMS for inquiry, analysis, and decision-making.

To provide transaction processing: Users input, update, and process data with transactions that generate information needed by managers and other users or by other departments.

2.3 Database Application

2.3.1 Client - Server Database Application

In 1980s the concept of Local Area Network (LAN) emerged to share computer peripherals like Printer, Plotters etc. But soon people felt the need to share databases too and this led to development of the multi user database applications on LAN. Eventually this led to the new style of multi user database processing called the client server architecture.

A client server application can be defined as one that has two parts: One part runs on the server, and the other part runs on workstations. The server side of the application provides security, fault tolerance, performance, concurrency, and reliable backups.

The client side provides the user interface and can contain empty reports, queries, and forms. In client/server computing, when a query is run, the server searches the database and sends only the rows that are a match to the client.

2.3.2 Database - Server Database Application

It's a specialized process that manages the database itself. The applications are clients to the database server and they never manipulate the database directly, but only make requests for the server to perform these operations.

Access is a relational database but it is not a database server. MS SQL, SQL Anywhere, DB2, Oracle, is both relational databases and database servers.

2.3.3 Database Using Internet Applications

Then come the database using Internet applications, which we see in majority of web sites today. Almost all major stores have the online stores using their own databases at the back end. Technologies like Active Server Pages, Cold Fusion and XML in particular being used to make database available through Internet.

2.3.4 Distributed Database Processing

The distributed database processing is one of the emerging database technology solutions and the essence of it is that all the Organizations data is spread over many computers- PCs, LAN servers, and mainframe- that communicate with one another as they process the database.

The goal of distributed database system is to make each user feel that he or she is the only user of the Organizations database and to provide same consistency, accuracy and timeliness that one would be getting if that person is the only person using the database.

However, the DBMS must periodically synchronize the scattered databases to make sure that they all have consistent data. For this various Replication Technologies too are used.

2.3.5 Object- Oriented DBMS (ODBMS)

It is also one of the emerging concepts of data storage and at this point we can only say that it is more complex way of storing data and more research on this concept is being done before it comes up as an Industry accepted technology.

Moreover ODBMS are expensive to develop. When you integrate database capabilities with object programming language capabilities, the result is an ODBMS. An ODBMS makes database objects appear as programming language objects in one or more existing programming languages.

The ODBMS extends the language with transparently persistent data, concurrency control, data recovery, associative queries, and other database capabilities

2.4 The Entity Relationship Model (E-R Model)

Entity Relationship Model was introduced by Peter Chen in 1976. It's important to note that Peter Chen set out the foundation of this model and over the period of time, this model has undergone a comprehensive change. The change has been brought by various people who worked on this model for all along this period. The E-R Model has been made part of various CASE Tools.

2.4.1 CASE Tools

Known as Computer Aided Software Engineering (CASE) tools. These tools are primarily used for design documentation and code generation. In simple layman terms, CASE is a tool, which aids a software engineer to maintain and develop software, much like a mechanic who uses the aid of his/her tools to maintain and develop vehicles, but obviously using respective techniques.

It is believed that CASE tools reduces cost as well as development time, and assures superior design consistency. Popular CASE tools are IEW, IEF, Visio, ER-Win etc. E-R Model is primarily used to represents the overall logical structure of the Database.

2.4.2 The most important elements of E-R Model

2.4.2.1 Entities

Entities are the principal data object about which information is to be collected. Entity is something that can be identified in the user's environment. It's something that the user wants to track. An entity is an object that exists and is distinguishable from other objects. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events, which have relevance to the database. Some specific examples of entities are Students, Instructors. An entity is analogous to a table in the relational model.

Entities of a given type are clubbed into Entity class. Example All students studying in an Institute attending various different classes. Thus the Students Entity class is the collection of all student entities. Thus an Entity class is a collection of entities and is described by the structure of the Entities in that class.

Whereas an Entity instance is representation of a particular Entity in the entity class such as Student Joe. There are usually many instances of an Entity in an Entity Class. Entity instance is analogous to a row in the relational table.

2.4.2.2 Attributes

Entities have Attributes. Attributes are also known as Entity properties or Domain. Attributes describe the entity of which they are associated. Examples of Attributes are Student Name, Student ID, and Class ID etc. Attributes is analogous to a column in the relational table.

It's an important assumption of an E-R Model that all instances of a given class have the same attributes.

Composite Attribute: Composite Attribute is one, which consists of group of attributes. Example of Composite Attribute is Address, which consists of group of attributes [Street, City, State, Zip]

Multi Value Attributes: When a particular Entity has more then one values. Example can be anyone's E-mail ID as person may have more then one E-Mail ID.

There can be also multi value composite attribute like a student can give his more then one phone number as a contact number. Now phone number is composite in the sense that it contains Area Code and Telephone Number and multi value in the sense that Student may provide more then one number as his or her Telephone number.

Identifiers Entity instances have identifiers, which are attributes that name or identify entity class. Like in student entity class student instance can be identified by student name or student id. Also faculty entity class instance can be identified by Faculty name or FacultyID.

The identifier usually contains one or more of the entities attributes. An identifier may either be unique or non unique. StudentID or FacultyID is most probably unique identifier but Student name or faculty name usually may not be a unique identifier as in an entity class more then one student or faculty have same name. Identifiers having more then one attributes are called composite identifiers.

2.4.3 Relationships

A Relationship represents an association between two or more entities. An example of a relationship would be:

- Students are assigned to projects
- Projects have subtasks
- Class manage one or more projects
- Relationships are classified in terms of degree, connectivity, cardinality, and existence.

E-R Model takes care of both Entity class relationships and Entity instance relationship. Like how STUDENT entity class related to INSTRUCTOR entity class and how a student Joe related to Instructor Tim or fellow student Smith.

The connectivity of a relationship describes the mapping of associated entity instances in the relationship. The values of connectivity are "one" or "many". The cardinality of a relationship is the actual number of related occurrences for each of the two entities. The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many.

A one-to-one (1:1) relationship is when at most one instance of an entity A is associated with one instance of entity B. For example, "employees in the company are each assigned their own cube. For each employee there exists a unique cube and for each cube there exists a unique employee. Example: Student and Book (Hash on student side and oval on book side)

A one-to-many (1:N) relationships is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example of 1:N relationships is a department has many employees and each employee is assigned to one department. Example: Student-Instructor: Each Instructor has many students to teach but all students have only one Instructor to learn from.

A many-to-many (M: N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. Example: Student can be assigned to no more than two projects at the same time or projects must be assigned to at least three students.

A single employee can be assigned to many projects; conversely, a single project can have assigned to it many employees. Here the cardinality for the relationship between employees and projects is two and the cardinality between project and employee is three. Many-to-many relationships cannot be directly translated to relational tables but instead must be transformed into two or more one-to-many relationships using associative entities.

2.5 The SQL Server Default Databases

When SQL Server is initially loaded it loads following 6 (4 System and 2 Learning databases) databases by default.

The Master Database: It's 9MB in size after installation. Master database is the key and most important database to run SQL Server. It contains pointer to the primary data file for every other database installed on the system, as well as key server wide information. The server wide information is usually error messages, login information, and system stored procedures etc.

The Model Database: It's 1.5 MB in size after installation. Model database is a template database, means that each time a database is created actually a copy of Model database is made. Subsequent changes of size etc are made in the template of Model database copied. Remember everything created inside Model database will be created in any new database created. Since Model is created to create each new database, no database can be smaller than the Model database size, which is 1.5 MB.

The Tempdb Database: It's 2.5 MB in size after installation. Tempdb database is where sorts, joins and other activities that require temporary space are performed. Tempdb database is reinitialized each time SQL Server is restarted.

The MSDB Database: It's 2 MB in size after installation. MSDB database is one that supports the SQL Server agent service, which includes storing information about jobs, alerts, events and replication. A history about all backups and Restore are kept in this database.

The PUBS Database: It's 2 MB in size after installation. PUBS database is a learning and sample database. Books online installed with SQL Server use PUBS database tables and other objects.

The North Wind Database: It's 4 MB in size after installation. The North wind database is an alternative learning and sample database. This database is in MS Access too

2.6 SQL Server Default Login ID

Default login for SQL Server is sa. Sa is a member of sysadmin fixed server role. At this point one needs to understand that sysadmin fixed server role member can do anything in SQL Server. This account can never be dropped.

2.7 A Brief on SQL Server Program Group

The Readme.txt File: The readme.txt file contains important information one should read before one install SQL Server, as well as the information that could not make it to manuals before shipping of the product.

Books Online: SQL Server Books Online is the primary reference source for the information on anything related to SQL Server. Books Online can be seen using IE 4.0 or later in SQL Server 7.0

Service Manager: Service Manager enables one to control the SQL Server related services on the Computer. Service Manager runs from the Task Bar. The services which can be controlled with SQL Service Manager are

- MSSQL Server Service
- SQL Server Agent Service

- MSDTC service
- MS Search Service

Client Network Utility: From here one can control the network library to be used when one try to connect to SQL Server using SQL Server client network utility. On Win NT Named Pipes is the default network library while for Win 9x, the default library is TCP/IP.

Server Network Utility: If client network utility is used to specify which network library to use to connect to SQL Server from client application, server network utility is used to specify which network libraries is SQL Server is listening on.

Query Analyze: It's the primary interface to run T-SQL queries or stored procedures. It can also be run from command prompt using ISQLW.exe.

T-SQL Script: T-SQL script is any kind of SQL commands that are stored and executed together. By default scripts have .SQL extension.

Enterprise Manager (EM): This is the primary graphical Administrative tool for SQL Server. Almost all things can be done in EM, which can be done using T-SQL commands. SQL Server is MMC snap-in. MMC is a common utility that MS and third party vendors can use as the common administrative interface to their respect products.

MSDTC Administrative Console: This lets one control distributed transactions. A transaction within a single SQL Server that spans two or more databases is actually a distributed transaction.

Performance Monitor: This is primarily used to monitor the SQL Servers performance.

Profiler: This utility enables one to monitor all the activity on the SQL Server. Can also be used to perform performance-tuning activities, such as examining the execution plan for SQL Server.

2.8 Creating Database in Sql Server

This can be accomplished using

- The SQL Server Enterprise Manager
- The Create Database statement
- The database creation Wizard

Since database needs file to physically store their data on the disk, so when database is created one need to specify at least two files one to store data and separate to store Transaction logs.

2.8.1 Create Database Statement Syntax (Simplified One)

```
Create database Dbase_name
[ ON [PRIMARY]
( [ NAME = logical_file_name, ]
FILENAME = 'os_file_name'
[, SIZE = size]
[, MAXSIZE = { max_size | UNLIMITED } ]
[, FILEGROWTH = growth_increment] ) [,...n]]
[ LOG ON
{ ( [ NAME = logical_file_name, ]
FILENAME = 'os_file_name'
[, SIZE = size]
[, MAXSIZE = { max_size | UNLIMITED } ]
[, FILEGROWTH = growth_increment] ) [,...n]
[,...n]} ]
[ FOR LOAD | FOR ATTACH ]
```

- Dbase_name- Is the name of the new database. Database names must be unique within a server and conform to the rules for identifiers. dbase_name can be up to 128 characters.
- ON Primary- Specifies of which file group this database will be member. Default file group is Primary.
- Name-This specifies the logical name to be used for reference within SQL Server.
- FileName- Specifies the filename and pathname where the data will be stored on the hard disk.
- Size- Specifies how big a database file should be. Can be expressed in MB or KB. Default is MB.
- Maxsize- Specifies the maximum size to which a database can grow. Can be expressed in MB or KB. This is an important feature as if one does not specify

then database keeps on growing dynamically and could fill the entire hard disk. Default is MB.

- Filegrowth- Specifies the parameter used for the autogrowth of the database. The FILEGROWTH setting for a file cannot exceed the MAXSIZE setting. Default if not specified is 1MB.
- Log On-Specifies where the transaction Log files are located and what size they are of
- For Load- This clause is supported for compatibility with earlier versions of Microsoft SQL Server. This parameter makes the database for dbo use only. Should be avoided in SQL Server 7.0.
- For Attach- Specifies that a database is attached from an existing set of operating system files.

2.9 T-SQL

Now we will see how to write simple SQL queries. For understanding T-SQL and to use examples below use Query Analyzer in SQL Server.

2.9.1 The Select Statement

One can use a SQL query to retrieve the data one wants from the database. A query has three parts: SELECT, FROM, and WHERE.

- SELECT: Tells SQL Server which columns one want to see as the result.
- FROM: Specifies which table or tables you want to use in the query.
- WHERE: Limits which rows one want to see.

The syntax isn't case sensitive, but generally people use uppercase and lowercase to distinguish the commands from the object names. A typical query looks like this:

```
SELECT col1, col2, col3......
FROM table1
WHERE col1 = search condition
```

2.9.2 Important Mathematical Functions

Mathematical functions enable one to return mathematical data using the syntax. Arithmetic operators can be used with data types: int, smallint, tinyint, numeric, decimal, float, real, money and smallmoney.

With arithmetic operators two types of precedence exists:

- Datatype Precedence: Arithmetic precedence exists when arithmetic operations are performed on different data types. In case of usage of different data types, the smaller data type is converted to higher data type. Like if we multiply smallint by an int, the result will be an int, but in case of money result will always be of data type money
- Operator Precedence: Used when multiple operators are used. It uses same math concept of BODMAS.

Mathematical functions are used as: "Select function_name(parameters)".

- Abs (numeric_expr): Provides absolute value.
  "Select Abs (-123)" gives [123]
- Ceiling (numeric_expr): Provides smallest integer greater than or equal to specified value.
  "Select ceiling(123.3), ceiling(-123.3)" gives [124,-123]
- Floor (numeric_expr): Provides largest integer less than or equal to specified value.
  "Select floor (123.3), floor(-123.3)" gives [123, -124]
- Round (numeric_expr, length): Provides numeric expression rounded to the specified length in an integer value.
  "Select Round (1234.56,1), Round ($123.67,1)" gives [1234.6,123.7]
- Square (float_expr): Provides square value.
  "Select Square(9)" gives (81)
- SQRT (float_expr): Provides square root value.
  "Select SQRT(9)" gives [3]
- POWER (numeric_expression, y) : Provides value of numeric expression to the power of y.
  "Select Power(2,3)" gives [8]

2.9.3 Important String Functions

String functions are used for manipulation of character data type

- + (expression, expression): Concatenates two or more strings.
  "Select lname +','+ fname +'.' As Name from employee"
- ASCII (char_expr): Provides ASCII code for left-most character.
  "Select ASCII ('A')" gives [65]
- Char (integer_expr): Provides character equivalent of ASCII code value.
  "Select Char(97)" gives [a]
- PATINDEX('% pattern%', expression): Returns starting position of first occurrence in expression.,
  "Select PATINDEX ('%bv%','ronvcbvvbc')"
- Lower (char_expr): Converts to lower case.
  "Select Lower ('ABCD')" gives [abcd]
- Upper (char_expr): Converts to upper case.
  "Select Upper ('abcd')" gives [ABCD]
- Left (char_expr, integer_expr) : Provides character string starting from the left and preceeding integer_expr character.
  "Select left('Seattle' ,2)" gives [se]
- Right (char_expr, integer_expr): Provides character string starting from the integer_expr character from Right.
  "Select Right ('Seattle' ,2)" gives [le]
- LTRIM (char_expr) : Returns data without leading blanks.
  "Select Ltrim ('   Computers')" gives [Computers]
- RTRIM (char_expr) : Returns data without trailing blanks.
  "Select Ltrim ('Computers   ')" gives [Computers]
- REPLACE ('string_expression1', 'string_expression2', 'string_expression3'): Replaces all occurrences of srting2 in string1 with string3.
  "Select Replace ('abcde', 'bc','oo')" gives [aoode]
- Reverse (char_expr) : Returns reverse of character expression.
  "Select Reverse('ABCD')" gives [DCBA]
- SUBSTRING(expression, start, length): Returns part of character string.
  "SELECT SUBSTRING('abcdef', 2, 3) as X" gives X =bcd

2.9.4 Date Functions

Date functions are used for manipulation of date data type. They are used as: Select date_function(parameter).

- Dateadd (datepart,number,date): Adds the number of dateparts to the date.
  "Select Dateadd (mm, 6, '1/1/01')" gives [2001-07-01]
- Datediff (datepart,date1,date2): Number of dateparts between two dates.
  "Select datediff (dd, '10/3/01', '10/9/01')"
- Getdate() :Returns current date and time in internal format.
  "Select Getdate()"
- Datepart (datepart, date): Returns integer value for specified datepart for date listed.

Table 2.1: Date Part Parameters

Datepart	**Abbreviation**
Day	DD,D
Day of year	DY
Hour	HH
Millisecond	MS
Minute	MI
Month	MM,M
Quarter	QQ
Second	SS, S
Week	WK
Weekday	DW
Year	YY, YYYY

Figure 2.4 Abbreviation of Date Part Parameters

2.9.5 System Functions

- Isdate (variable) :Checks for valid date.

  "Select Isdate('23')" Returns 1 if valid else returns 0 OR Select Isdate('12/1/01')
- IsNumeric: Checks for valid numeric format. Returns 1 if valid else 0.

  "Select IsNumeric(12)"
- IsNull (expression, value) : Returns specified value in place of NULL.

  "Select Substring (title, 1, 15) AS Title, type AS Type, IsNull(price, 0.00) AS Price From titles"

2.9.6 Comparison Operators

- One can use comparison operators like >,<, >=, =<, =, <>, !=, !>, !<, ().

  "SELECT emp_id,lname, fname FROM employee WHERE pub_id='0877'"
- One can retrieve rows based on a range of value using the BETWEEN keyword

  "Select lname, emp_if FROM Employee WHERE hire_date between '10/1/92' AND '12/31/92'"

  Smaller value must come first in BETWEEN clause. Also one must enclose range in quotes if its character data types or date data types.

2.9.7 List Operators

- IN: One can use IN Keyword to match rows in a list.

  SELECT Column_list FROM table_name WHERE column_name [NOT] IN (value_list)

  "SELECT emp_id, lname, fname FROM employee where pub_id IN('0877','9990')"
- LIKE: It is used with character and date data.

  "SELECT column_name FROM table_name WHERE column_name [NOT] Like 'string'"

2.9.8 Wildcards

- % : String of zero or more characters
- _ : Single Character
- [] : Single character within specified range
- [^] : Single character not within specified range

"SELECT title_id, title FROM titles WHERE title LIKE '%computers%'"

Wildcard characters when used with Like keyword are enclosed in a single quotation mark.

2.9.9 Unknown Values

Null is equivalent to value "unknown". IS NULL and IS NOT NULL operators are used . Syntax is: SELECT column_list FROM table_list WHERE column_name IS [NOT] NULL. An example is shown below:

"SELECT title_id, title FROM titles WHERE title_id IS NOT NULL"

2.9.10 Using Distinct To Eliminate Duplicate Information

One can eliminate duplicates using Distinct keyword in the SELECT clause. Syntax is: SELECT DISTINCT column_list FROM table_Name Where search_condition. An example is shown below:

"SELECT DISTINCT city FROM authors"

2.9.11 Using ORDER BY Clause

ORDER BY clause is used in SELECT statement to sort data. Syntax is: SELECT column_name From Table_List [order by column_name[ASC|DESC]]. An example is shown below:

"SELECT title_id, au_id FROM titleauthor ORDER BY title_id, au_id"

Default Order is ascending. Since sort order being used by SQL Server can make difference in the ORDER BY clause, one should check default sort order of SQL Server using sp_helpsort. ORDER BY clause can not be used on text or image columns.

2.9.12 Aggregate Functions

Aggregate functions can return summary values for an entire table or for group of rows in a table. Aggregate functions are usually used in conjunction with the GROUP BY clause and are used in HAVING Clause or in the column_list.

- AVG: Returns average of the values in the numeric expression.
  "SELECT AVG(ytd_sales) FROM Titles"
- COUNT(*): Returns number of selected rows.
  "SELECT Count(*) from employee" Returns total number of rows in a table.
- MAX: Returns highest value in the expression.

"SELECT MAX(ytd_sales) FROM titles"

- MIN: Returns lowest value in the expression.
  "SELECT MIN (ytd_sales) FROM titles"
- SUM: Returns total of values in the numeric expression.
  "SELECT SUM(qty) FROM sales"

2.9.13 Group By and Having Clause

The group by clause summary data that meets the WHERE clause criteria to be returned as single row. The HAVING clause set the criteria to determine which rows will be returned by the GROUP BY clause.

"SELECT title_id, count (title_id) as Number_of_Authors From Titleauthor GROUP BY title_id Having count (title_id)>1"

OR

"SELECT title_id,ytd_sales FROM Titles WHERE (ytd_sales>=4000) GROUP BY title_id,ytd_sales"

HAVING clause has same effect on the GROUP BY clause as the WHERE clause has on the SELECT statement. GROUP BY clause must contain all nonaggregate columns from the SELECT column_list. HAVING clause criteria columns must return only one value.

2.9.14 Implementing Joins

Joins are used to retrieve data from two or more tables. The results appear as a single table with columns from the entire table specified in the SELECT column_list and meeting the search criteria. In nutshell Joins connect two or more tables based on a join condition and produce results as new table with the rows that satisfy the join condition. We will probe Joins using both SQL Server syntax and ANSI syntax.

Inner Join: There can be two types of Inner Joins. One is Equi-Join and other is Natural Join. Using Pubs consider tables:

- Publishers (pub_id, pub_name, city, state, country) Primary Key (pub_id)
- Pub_info(pub_id, logo, pr_info) Primary Key (pub_id)
- Authors (au_id, au_lname, au_fname, phone, address, city, state, zip, contact) Primary Key (au_id)

Equi-Join: In Equi-Join Column values are compared for equality and redundant columns are displayed as columns in the result set.

SQL Server syntax:

```
"SELECT * FROM publishers, pub_info
WHERE  publishers.pub_id=pub_info.pub_id"
```

ANSI syntax:

```
"SELECT * FROM Publishers INNER JOIN pub_info
ON publishers.pub_id = pub_info.pub_id"
```

Natural Join: In Natural Joins redundant columns are not displayed twice.

SQL Server syntax:

```
"SELECT p. *, p1.logo, p1.pr_info FROM publishers p, pub_info p1
Where p.pub_id = p1.pub_id"
```

ANSI syntax:

```
"SELECT p.*, p1.logo, p1.pr_info FROM publishers p
INNER JOIN pub_info        p1 ON p.pub_id = p1.pub_id"
```

Cross Join or Unrestricted Join: Returns combination of all rows of all tables in join as the result set. Each row of one table is joined with each row of another table.

SQL Server syntax:

```
"SELECT p.pub_name, p1.pr_info  FROM publishers p, pub_info p1"
```

ANSI syntax:

```
"SELECT p.pub_name, p1.pr_info
FROM publishers p CROSS JOIN pub_info  p1"
```

Outer Join: Restricts rows from one table while allowing all rows from another table as a result set. Usually used for orphan records. Outer joins can be a left, right, or full outer join. Outer joins are specified with one of the following sets of keywords when they are specified in the FROM clause: LEFT JOIN or LEFT OUTER JOIN.

The result set of a left outer join includes all the rows from the left table specified in the LEFT OUTER clause, not just the ones in which the joined columns match. When a row in the left table has no matching rows in the right table, the associated result set row contains null values for all select list columns coming from the right table.

"SELECT a.au_fname, a.au_lname, p.pub_name FROM authors a

LEFT OUTER JOIN publishers p ON a.city = p.city

ORDER BY p.pub_name ASC, a.au_lname ASC, a.au_fname ASC"

If Inner Join were used in above query then we would have got only the results of authors name whose city was same as publishers and not those authors who lived in city to which no publisher belonged to. In this case we got name of all the authors

Self Join: Correlates rows of a table with other rows in the same table. A table can be joined to itself in a Self Join. For example, you can use a Self Join to find out the authors in Oakland, California who live in the same zip code area. Because this query involves a Join of the authors table with itself.

"SELECT au1.au_fname, au1.au_lname, au2.au_fname, au2.au_lname

FROM authors au1 INNER join authors au2 ON au1.zip = au2.zip

WHERE au1.city='Oakland' AND au1.state = 'CA' AND au1.au_id < au2.au_id"

2.9.15 Data Modification with Queries

Inserting Records: The basic Insert statement adds one row at a time to a table. Variations of Basic Insert statement allows one to add more than one row at a time by selecting data from another table or by executing a stored procedure, but in that case one need to know the structure of the table.

While Inserting data in a table its useful to know:

- Number of columns in a table
- The data type of each column,
- Column names,
- Column properties like Identity or timestamp [A database-wide unique number in a table. There can be only one unique timestamp column in a table.

Using the INSERT statement syntax is: INSERT INTO Table Name(Column Names) VALUES (Column Names). All character values are to be enclosed in quotation marks and numeric without quotations. But if one need to supply values in a different order, or if one do not want to supply an explicit value for a column one can use INSERT statement that has a list of column names before the VALUES clause. An example shown below:

"INSERT INTO mypublishers (state, pub_id) VALUES ('WA','9999')"

Deleting Records: This data modification statement allows one to remove one or more rows from a table. Syntax is: Delete From TableName Where Condition. An example shown below:

"Delete mypublishers Where pub_name='New Moon Books'"

Updating Records: Update is used to change the values within an existing row. Syntax is: Update Table Name Set Column Name Where Condition. An example shown below:

"Update publishers SET city='seattle'
WHERE pub_name= 'Five Lakes Publishing'"

CHAPTER THREE: ASP .NET AND C# LANGUAGE ON VISUAL STUDIO .NET 2000 PLATFORM

3.1 What Is .NET?

When .NET was announced in late 1999, Microsoft positioned the technology as a platform for building and consuming Extensible Markup Language (XML) Web services. XML Web services allow any type of application, be it a Windows- or browser-based application running on any type of computer system, to consume data from any type of server over the Internet. The reason this idea is so great is the way in which the XML messages are transferred: over established standard protocols that exist today. Using protocols such as SOAP, HTTP, and SMTP, XML Web services make it possible to expose data over the wire with little or no modifications to your existing code. Figure 3.1 presents a high-level overview of the .NET Framework and how XML Web services are positioned.

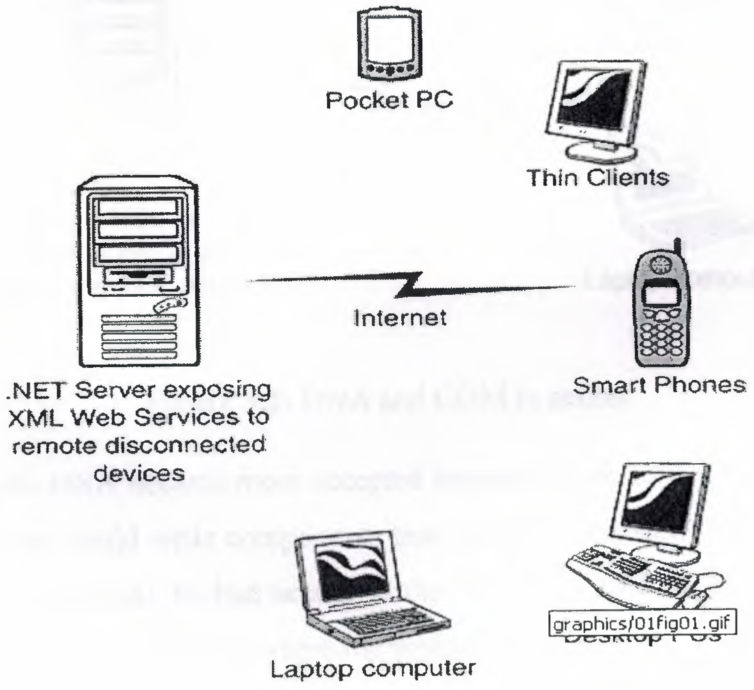


Figure 3.1: Stateless XML Web services model.

Since the initial announcement of the .NET Framework, it's taken on many new and different meanings to different people. To a developer, .NET means a great environment for creating robust distributed applications. To an IT manager, .NET means simpler deployment of applications to end users, tighter security, and simpler management. To a CTO or CIO, .NET means happier developers using state-of-the-art development technologies and a smaller bottom line. To understand why all these statements are true, you need to get a grip on what the .NET Framework consists of, and how it's truly a revolutionary step forward for application architecture, development, and deployment.

3.1.1 Windows DNA and COM

Writing distributed Internet applications became easier as the model of COM services that Windows servers could provide became more stable and widespread. You could write an Active Server Pages (ASP) application and access methods, properties, and events through the object model of components running inside of COM+ services on remote machines.

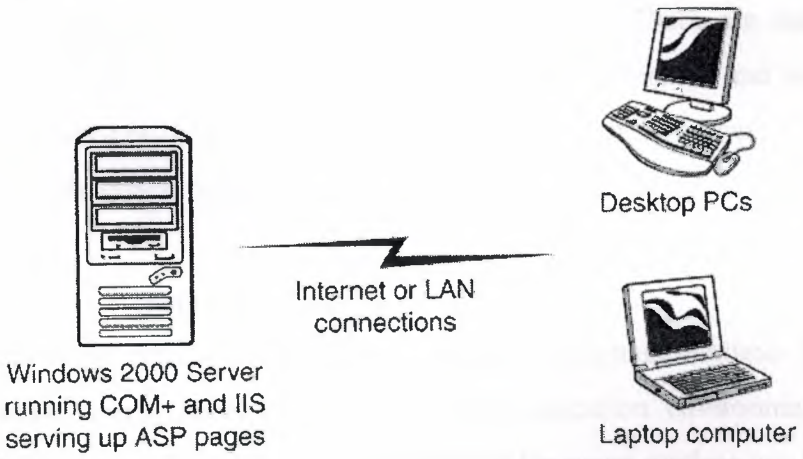

Desktop PCs
Internet or LAN connections
Windows 2000 Server running COM+ and IIS serving up ASP pages
Laptop computer

Figure 3.2: DNA and COM in action.

Windows DNA became more accepted because of the ease with which a Visual Basic 6 developer could write components that could be accessed from any other type of application, as long as he had access to the Windows 2000 server that the COM+ services were running on. This is where the problems begin.

If you provide data to outside vendors in an application, you must write the user interface and code to allow them access to what they need. There's no simple way to expose methods, or allow other applications to call methods, on your servers. You have to open up security and give them the keys to the farm, which isn't what IT managers are likely to do. If you want to maintain multiple versions of a component, you are in a very bad way with COM. Because COM makes heavy use of the Registry and doesn't allow for a simple versioning policy, you're essentially maintaining the same component forever. You're constantly adding new features to it, but leaving the old stuff in. This is one of the big rules of COM: Thou shall not change any interfaces to your components. Doing so makes for a huge headache in deployment. If you change an in or out parameter on a method, you've broken the functionality of the component. That means all the components must be recompiled to restore the correct interfaces that the caller expects. After a component is deployed, how do you easily scale it across machines while maintaining any state data that the code expects? This isn't a trivial problem, and companies have spent millions of dollars writing state machines to handle the scalability problems that come with COM.

All these issues are solved with the .NET Framework. The services provided by the .NET Framework enable us to expose methods over HTTP and SOAP through XML Web services. The Windows Registry is not used in .NET, which eliminates the DLL Hell of the past and gives us a strong versioning policy. There are many ways to maintain state data, so we can scale applications across processors and across servers with no worry about crashing the applications running on those servers. This all starts with the common language runtime and the base class libraries.

3.1.2 The Common Language Runtime

At the heart of the .NET Framework is the common language runtime. The common language runtime is responsible for providing the execution environment that code written in a .NET language runs under. The common language runtime can be compared to the Visual Basic 6 runtime, except that the common language runtime is designed to handle all .NET languages, not just one, as the Visual Basic 6 runtime did for Visual Basic 6. The following list describes some of the benefits the common language runtime gives you:

- Automatic memory management
- Cross-language debugging
- Cross-language exception handling
- Full support for component versioning
- Access to legacy COM components
- XCOPY deployment
- Robust security model

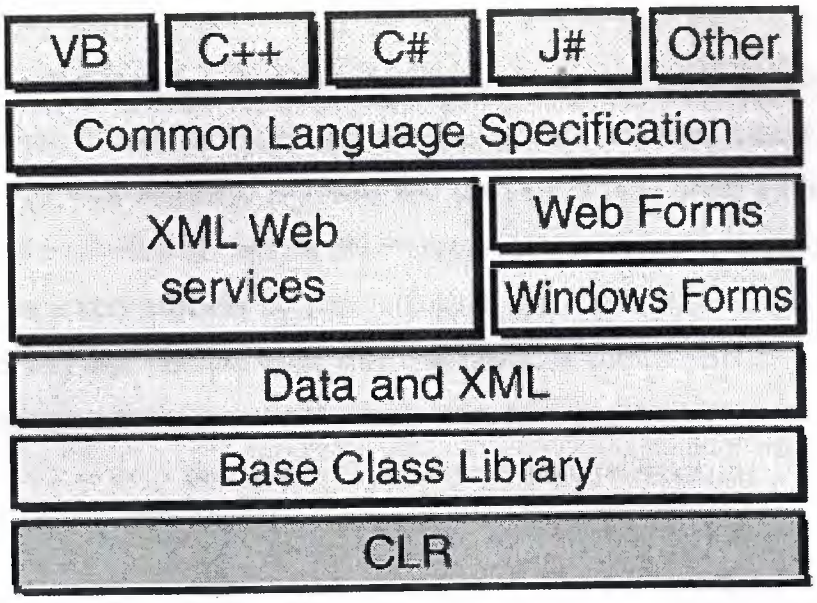


Figure 3.3: Common Language Scheme

3.1.3 Inside the Common Language Runtime

The common language runtime enables code running in its execution environment to have features such as security, versioning, memory management, and exception handling because of the way .NET code actually executes. When you compiled Visual Basic 6 forms applications, you had the ability to compile down to native node or p-code.

When you compile your applications in .NET, you aren't creating anything in native code. When you compile in .NET, you're converting your code—no matter what .NET language you're using—into an assembly made up of an intermediate language called Microsoft Intermediate Language (MSIL or just IL, for short). The IL contains all the information about your application, including methods, properties, events, types, exceptions, security objects, and so on, and it also includes metadata about what types in your code can or cannot be exposed to other applications. This was called a type library in Visual Basic 6 or an IDL (interface definition language) file in C++. In .NET, it's simply the metadata that the IL contains about your assembly.

Understanding the process of compilation in .NET is very important because it makes clear how features such as cross-language debugging and exception handling are possible. You're not actually compiling to any machine-specific code—you're simply compiling down to an intermediate language that's the same for all .NET languages. The IL produced by J# .NET and C# looks just like the IL created by the Visual Basic .NET compiler. These instructions are the same, only how you type them in Visual Studio .NET is different, and the power of the common language runtime is apparent.

When the IL code is JITted into machine-specific language, it does so on an as-needed basis. If your assembly is 10MB and the user is only using a fraction of that 10MB, only the required IL and its dependencies are compiled to machine language. This makes for a very efficient execution process. But during this execution, how does the common language runtime make sure that the IL is correct? Because the compiler for each language creates its own IL, there must be a process that makes sure what's compiling won't corrupt the system. The process that validates the IL is known as verification.

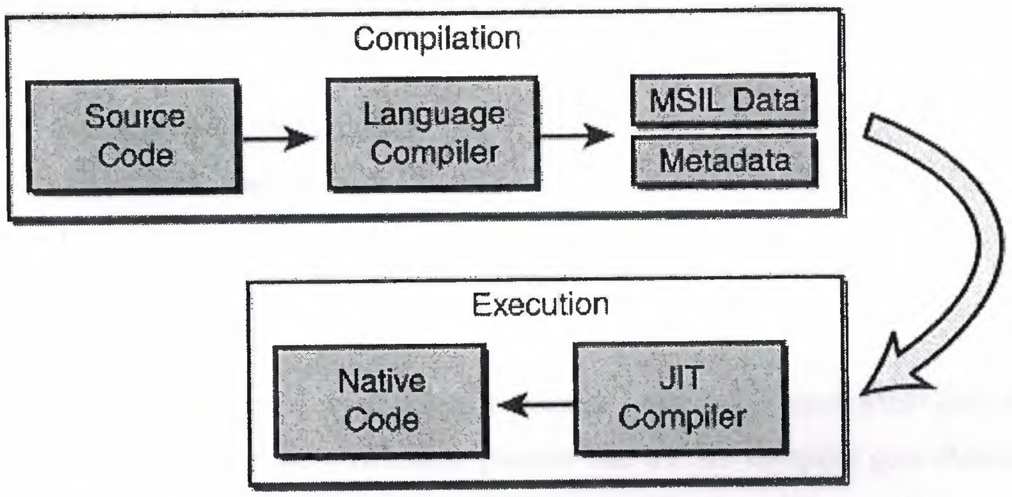


Figure 3.4: The JIT process and verification.

3.1.4 Understanding the Common Language Specification

The CLS describes the concrete guidelines that make a .NET language compliant with the common language runtime. That doesn't mean a .NET language can't have language-specific features, but it does indicate that to be considered a .NET language, the language must comply with the set of requirements set forth in the CLS. All features added to a .NET language and that aren't part of the CLS won't be exposed to other .NET languages at runtime.

If your code is fully CLS compliant, it's guaranteed to interoperate with all other components written in any .NET language. Certain languages, such as C#, attempt to accommodate developers moving from C and C++ with the similarity of their syntaxes. Because C# attracts such developers, it includes functionality familiar from their native languages, such as pointers and code access to unsafe memory blocks. This functionality is not CLS compliant and won't be accessible by other .NET languages, but it's allowed by the common language runtime and the language-specific compilers. To make sure that your code is CLS compliant, compilers such as C# include checks for non-CLS-compliant code through the use of attributes. If you apply the CLSCompliantAttribute attribute to a class or method in your code and the code isn't CLS compliant, an error occurs and the compile fails. The following code demonstrates how to apply the CLSCompliantAttribute attribute in your code:

```
using System;
[assembly: CLSCompliantAttribute(true)]
[CLSCompliantAttribute(true)]
public class Class1
{
  public void x(UInt32 x){}
  public static void Main( )
  {
   }
}
```

In this case, the code won't compile because unsigned integers aren't part of the CLS. The second part of the verification process that the JIT compiler goes through to make sure that your code executes correctly is the verification of types. All types used in .NET must conform to the CTS.

3.1.5 Understanding the Common Type System

The CTS sets forth the guidelines for data type safety in .NET. In the past, there were no rules for type safety across execution runtimes, hence the general protection fault (GPF) and blue screen of death errors that could occur when running applications. The culprit behind those meltdowns was the overlapping of memory by data types. This was a common occurrence in Windows 3.1, Windows 95, and Windows 98. When a Visual Basic developer deployed a new application, fingers had to be crossed to make sure that the data types and memory access between the newly installed DLLs and the existing ones on the system mingled happily. Most of the time they did, but when they didn't, errors occurred.

In .NET, the CTS defines types and how they can act within the bounds of the common language runtime. There are two type classifications in .NET: value types and reference types.

Value Types: Value types directly contain the data you assign them. They're built into the common language runtime and derive directly from the base System.Object type. Examples of value types are primitive types, structures, and enumerations. Primitive types can be further broken down into numbers, such as Boolean, byte, short, integer, long, single, double, decimal, date, and char.

Reference Types: Reference types don't directly contain any data; rather, they point to a memory location that contains the actual data. Reference types are built into the common language runtime and derive directly from the base System.Object type. Some examples of reference types are strings, classes, arrays, delegates, and modules.

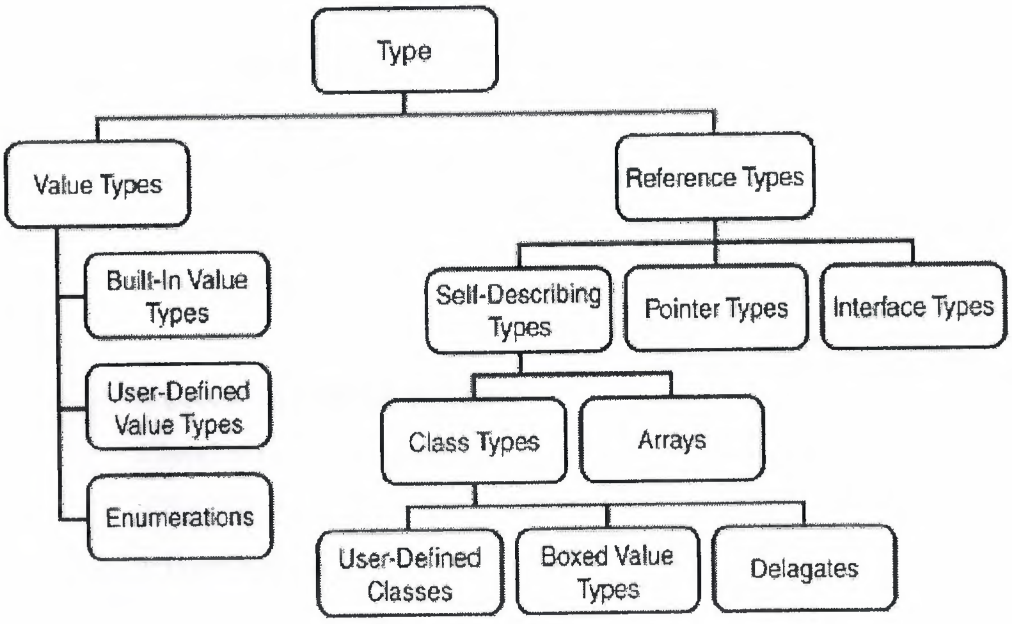


Figure3.5: Data types in .NET

C# Code Example

```
using System;
namespace cSharp_ValueReference
{
        class Class1
        {
                static public int x;
                 [STAThread]
                static void Main(string[] args)
                {
                        x=4;
                        int y;
                        y = x;
                        x=0;
                        // Since each Value type contains its own data,
                        // modifying the variable X after setting Y to the value
                        // of X does not affect either variable
                        Console.WriteLine(x);
                        Console.WriteLine(y);

                        // Create an instance of Class2
                        Class2 ref1 = new Class2();
                        // Set the refValue of this instance to 5
                        ref1.refValue=5;

                        // Create an object reference to the ref1 class
                        Class2 ref2 = ref1;
                        // Set the refValue of the object
                        ref2.refValue=10;

                        // Notice how the results are the same, even
                        // though you set re1.refValue to 5, the reference
                        // to this memory was overridden by the value of 10
                        Console.WriteLine(ref1.refValue);
                Console.WriteLine(ref2.refValue);
                Console.ReadLine();
            }
        }

        class Class2
        {
            public int refValue;
        }
}
```

In example, the values of the value type variables X and Y are 0 and 4, whereas the values of the reference types ref1 and ref2 are both 10. Because the reference type points to the same memory allocation for the initial object ref1, the value for all variables set to an instance of that object is always the last value assigned.

3.1.6 The .NET Framework Class Library

The second most important piece of the .NET Framework is the .NET Framework class library (FCL). As you've seen, the common language runtime handles the dirty work of actually running the code you write. But to write the code, you need a foundation of available classes to access the resources of the operating system, database server, or file server. The FCL is made up of a hierarchy of namespaces that expose classes, structures, interfaces, enumerations, and delegates that give you access to these resources.

The namespaces are logically defined by functionality. For example, the System.Data namespace contains all the functionality available to accessing databases. This namespace is further broken down into System.Data.SqlClient, which exposes functionality specific to SQL Server, and System.Data.OleDb, which exposes specific functionality for accessing OLEDB data sources. The bounds of a namespace aren't necessarily defined by specific assemblies within the FCL; rather, they're focused on functionality and logical grouping. In total, there are more than 20,000 classes in the FCL, all logically grouped in a hierarchical manner.

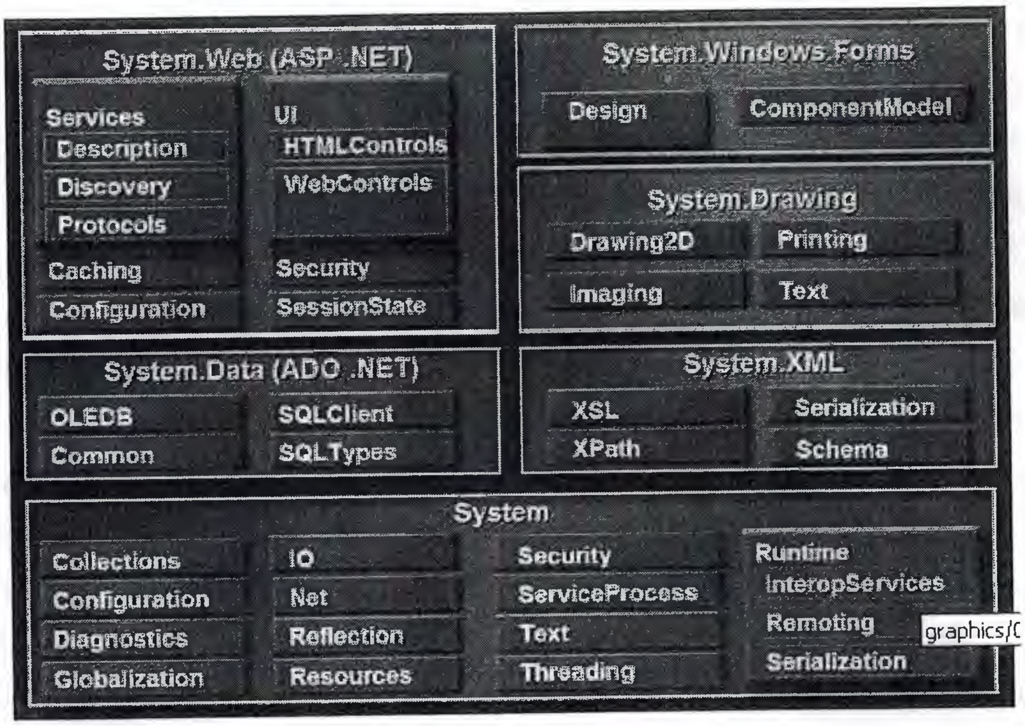


Figure 3.6: The .NET Framework class library grouped by namespaces logically.

To use an FCL class in your application, you use the Imports statement in Visual Basic .NET or the using statement in C#. When you reference a namespace in Visual Basic .NET or C#, you also get the convenience of auto-complete and auto-list members when you access the objects' types using Visual Studio .NET. This makes it very easy to determine what types are available for each class in the namespace you're using. As you'll see over the next several weeks, it's very easy to start coding in Visual Studio .NET. The following code adds a reference to the data classes in both Visual Basic .NET and C#.

3.1.7 .NET Servers and the Future of .NET

The designers of the .NET Framework put much thought into how distributed computing should work. It seems that .NET is the next killer app, but to make the .NET Framework a widespread success, actual servers must be built using the .NET Framework. Currently, there are no true .NET servers. There are servers that take advantage of the common language runtime and its managed execution environment, but most servers from Microsoft today still run under COM and unmanaged code.

Commerce Server 2002 is positioned as a .NET server for e-commerce, and applications you design with it can be completely written using Visual Basic .NET or

C#, but the underlying infrastructure of Commerce Server is still based on COM. Because rewriting server applications is a truly monumental task, the move to completely .NET servers could take several years. Along the way, there'll be servers such as Commerce Server 2002 that are half managed code and half unmanaged code. From a developer's viewpoint that's fine, because you don't want to write ASP and Visual Basic 6 code for server products while the rest of your distributed application development is in a .NET language.

Currently, Microsoft seems to be positioning server products as .NET Enterprise Servers if they can integrate XML Web services into their existing infrastructure. For example, SQL Server 2000 certainly isn't written in managed code, but there are add-ons to SQL Server 2000 that enable you to expose stored procedures as XML Web services. The SQL Server Notification Service is a .NET add-on that allows notification to .NET applications if certain events trigger in SQL. BizTalk server's purpose in life is the orchestration and automation of complex business processes, and it's positioned as a .NET server because of its capability to consume XML Web services. The following Microsoft server products are considered .NET Enterprise Servers because of their capability to at least interact with a distributed environment such as the Internet and have some relationship with the .NET Framework concepts:

- Internet Security and Acceleration Server
- Application Center 2000
- Commerce Server 2000 and Commerce Server 2002
- BizTalk Server 2000 and BizTalk Server 2002
- SQL Server 2000
- Exchange Server 2000
- Host Integration Server 2000

In my opinion, the fact that a .NET server is truly running under the common language runtime is not a deal breaker. For .NET to get to the next step, it must run on other operating systems, not just the Windows family of desktop and server operating systems. Currently, the Mono project is a grass-roots move to port the .NET Framework class library to the Linux operating system. That means the code you're writing now for Windows will also eventually run under Linux and, hopefully, Unix as well. You can learn more about the Mono project and where it currently is in the development process at http://www.go-mono.org. It would be a huge step forward if .NET were ported to the

Macintosh operating system also. Although the Mac is still a small percentage of the overall market in desktop PCs, its incompatibility with Windows creates headaches for application developers. There needs to be consistency across platforms eventually.

Moving into the future with .NET, the sky seems to be the limit. This isn't necessarily because Microsoft is going to think of some great new thing to add to the .NET Framework, even though it most likely will, but it has to do with computing in general and the general infrastructure of our daily lives. As every household and business installs high-speed data access, and as computers become faster and cheaper, the applications you write will have a greater influence on how people look at what computer programs can do. You aren't bound to single servers anymore. Writing truly distributed and scalable applications is very easy because of the groundwork laid out by the .NET Framework. You can begin to look at the code you write not as blocks of modules running on a Windows 2000 Server, but as distributed objects that you can reuse in multiple applications across an enterprise simply by plugging them into an XML Web service. The future of .NET is the concept of a true distributed environment.

3.2 ASP.NET Applications

ASP.NET is an object-oriented, event-driven development platform for writing Web-based applications. Before .NET, Active Server Pages was the core Microsoft technology for developing applications that ran through the browser. ASP was a great platform, and it truly revolutionized the way Web applications were written, but it had lots of room for improvement. With ASP.NET, the gap between writing Windows-based applications and Web-based applications has been closed.

Because ASP.NET is based on the .NET Framework, the same classes in the Framework class library (FCL) are available to all .NET-based applications. That means the same coding model that you use to write Windows Forms applications is used to write ASP.NET applications. It also means you can write ASP.NET applications in any .NET language.

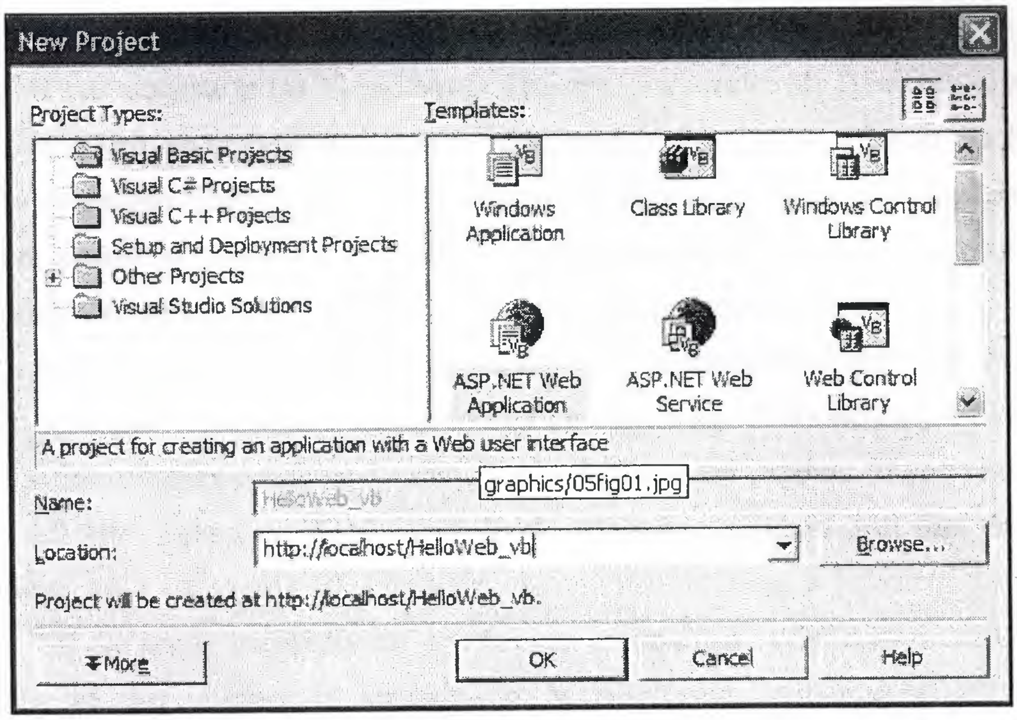
One of the drawbacks to writing ASP applications is the scalability issue. Because ASP pages are written in script, the code must be interpreted each time a page is accessed. To improve performance, developers write complex caching schemes, use different session state handling optimizations to improve page throughput, and move code into compiled COM components to increase performance—that all changes with

.NET. All ASP.NET applications are compiled. There's no interpreted script of server-side code, and the ASP.NET runtime is a multithreaded asynchronous application, so the core infrastructure is more scalable than ASP. The following bullets highlight some of the major improvements and benefits that ASP.NET gives you, and should also serve as a list of why ASP.NET is so much better than ASP:

- Using the various caching methods in ASP.NET, you can drastically improve application performance with a simple page directive—not any complex code.
- Using the Web.Config configuration file, you can implement robust page- and directory-level security without writing code for each page.
- Using Visual Studio .NET, you have the same Code Editor features that Windows Forms applications have, including auto-complete and auto-list members.
- Debugging ASP.NET applications is just like debugging Windows applications. All the great debugging features come from Visual Studio .NET, not the language or type of application you're writing.
- The event-driven model familiar to Visual Basic 6 developers and Windows Forms development is the same in ASP.NET, which makes application development more logical and very rapid.
- Visual Studio .NET provides a vast array of controls, including validation controls, grid controls, and list controls—all of which support data binding.
- ASP.NET provides several ways to implement session state, including page level, session level, and application level. The state data can be stored in memory, in cookies, in a Windows Service process, or in a SQL Server database, giving you great options for designing scalable applications.
- ASP.NET separates the visual pages from the page logic, giving you increased flexibility in how you design and develop applications.
- ASP.NET applications are 100% compiled.

3.2.1 Hello ASP.NET!

To create your first ASP.NET application, start Visual Studio .NET. At the Start Page, click the New Project button. In the New Project dialog, select either Visual Basic Project or C# Project, depending on what language you're writing in. From the Template pane, select the ASP.NET Web Application template. Change the name of the application to http://localhost/HelloWeb_vb or http://localhost/HelloWeb_cs, depending on what language you're coding in. Click the OK button to create the application.



A project for creating an application with a Web user interface

graphics/05fig01.jpg

Name: HelloWeb_vb

Location: http://localhost/HelloWeb_vb

Project will be created at http://localhost/HelloWeb_vb.

Figure 3.7: Selecting the ASP.NET Web Application template.

When you create an ASP.NET application, Visual Studio creates the virtual Web directory at the location you specify in the name of the project. In this case, the application was created at localhost, which is the default name for the local Web server running on your machine. If you want to create an application on another server, you specify the Web address and project name, and the application is created there.

You're now in the Visual Studio .NET designer for ASP.NET applications. You'll notice the same look and feel you saw when you were designing Windows Forms applications. The Solution Explorer displays the components of your project, the Properties window, and the Toolbox. All the windows you learned about over the last few days are available. The design environment is the same no matter what type of application you're developing. To understand where the files are created, navigate to the \inetpub\wwwroot\helloweb_cs or \inetpub\wwwroot\helloweb_vb directory.

When you create new applications in ASP.NET, all the core application files are created directly on the Web server. This is different from writing ASP applications using Visual InterDev, which creates a local copy of files and then creates a copy on the

Web server. You use the tools in Visual InterDev to release and refresh copies of files from the Web server. It's extremely important to understand that in ASP.NET and Visual Studio .NET, files are created directly on the Web server and there's no local copy.

If you navigate to the Visual Studio Projects folder under My Documents, you'll see a HelloWeb_vb or HelloWeb_cs directory, containing only the solution file (with the .sln extension). The solution file contains information about where the actual project files are located, so if you're ever having trouble opening a Web project that you didn't create, you can edit the .sln file to determine where the project files are located.

In the Solution Explorer, the files displayed in the hierarchical tree view are only the top-level files in the inheritance hierarchy for each class. That's why more files are listed in the wwwroot directory than you see in the Solution Explorer. If you click the Show All Files button on the Solution Explorer toolbar, you can drill into the file hierarchy.

All the server events for an ASPX page are handled in the code-behind files, whereas the user interface is contained in the ASPX file itself. Each ASPX page contains page directive attributes that set processing information on the page and indicate the inheritance hierarchy for the page. If you click the HTML button in the lower left of the Web Forms Designer, you're switched to the HTML for the ASPX file. Notice the page directives highlighted across the top of the page.

The CodeBehind attribute contains the WebForm1.aspx.vb file, which is the class file under WebForm1.aspx in the Solution Explorer. The Inherits attribute specifies the Page class for WebForm1 to inherit. This is different from Windows Forms, where a single class file inherits a class that contains the designer information. A WebForm has a class file that's physically separated from the designer file, which is actually the ASPX page. The ASPX page has information about what class file it should use to handle its events. The hierarchy of the files in the Solution Explorer is defined by the Page directive attributes for each file in the Solution Explorer.

If you click the Design button in the lower-left corner of the HTML designer, you're taken back to the design surface for the ASPX page. To get to the code-behind file, you can double-click the ASPX page (as you do for a Windows Form), you can press the F7 key, or you can double-click the class file in the Solution Explorer. When you get to the code-behind class.

The class file inherits the System.Web.UI.Page class. By inheriting this class for the page, all the Web-related methods, properties, and events are available to the code-behind class file.

When an ASP.NET application is compiled, the code-behind files for each ASPX page are compiled into a single assembly with a DLL extension and are placed in the Bin directory. Each time an ASPX page is called, the ASP.NET runtime on the server creates a class for the ASPX page. The created DLL uses the information in its Page directive attributes to determine where to get its events. The combination of the ASPX page's DLL and the solution's DLL make up the HTML that's outputted to the end user's Web browser.

Before we go on to the next section and begin adding controls and code to the WebForm1.aspx page, the files that make up a solution will be shown below:

Webform1.aspx: The visual part of the Web Form. You can modify the HTML to pass the user interface of a Web form or drag controls from the Toolbox onto the Web Form.

WebForm1.aspx.cs: Code-behind class file that the ASPX file inherits its functionality from.

Web.Config: XML-based configuration file for the project. You can set properties on security, caching, state, and tracing information in the Web.Config file.

Global.asax: Similar to the Global.asa file from ASP, the Global.asax file contains application-level events for an ASP.NET project.

Styles.css: Default style sheet for a project. You can double-click .css files to edit them in the Style Sheet Designer.

AssemblyInfo: Class file that contains assembly-specific information for the project's assembly output.

.vdisco: Discovery file for XML Web services in the application. You learn more about discovery files on "XML Web Service in .NET," when you learn about Web services.

3.2.2 Adding Controls to a Web Form

Adding controls to a Web Form is exactly like adding controls to a Windows Form. You visually design a page by dragging predefined controls from the Toolbox onto the Web Form. Server controls and Web controls are equivalent terms. The term server controls is more common in referring to controls that provide server-side event functionality.

When programming in ASP.NET, you have a choice of what types of controls you can use on a Web Form. There are HTML controls and server controls. HTML controls are the same HTML-tag-based controls you use in ASP or a regular HTML page. Server controls offer a richer user interface, and have an object model that you can access in the code-behind class files for an ASPX page.

To get an idea of the controls you can use, click the HTML Controls tab on the Toolbox, and then click the Web Form Controls tab on the Toolbox.

CHAPTER FOUR: WEB APPLICATIONS OF FAST – FOOD SALE AND RESTAURANT SALE MANAGEMENT

4.1 Definition of the Work:

Shopping on internet mentioned as e-commerce is very popular way to buy-sell products. Most of the componies are provided to come into contact with their customers via internet nowadays and the rate of internet sales over total one increasing day by day.

This project is prepared to cover up the need of food shopping on internet and managing the sales from restaurants. So, it contains two sections, one for internet shopping and another for restaurants. Two sections are investigated in different titles along the chapter.

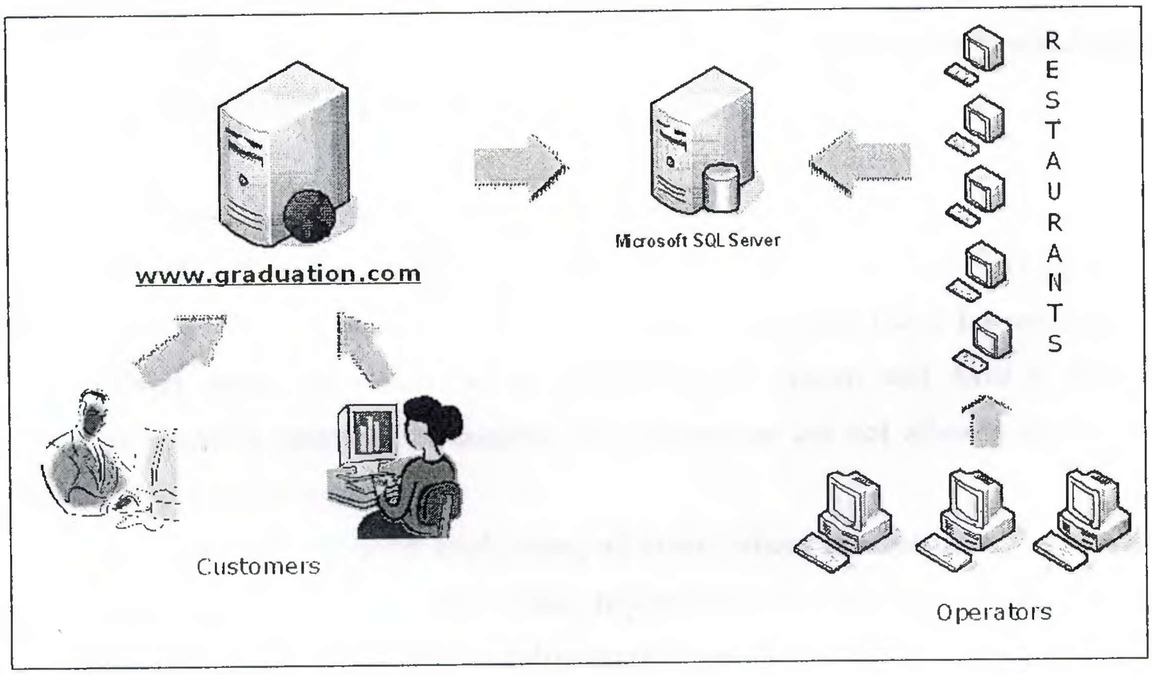


Figure 4.1: Working Scheme of the System

4.1.1 Explanation of Fast – Food Sale Web Application

Customer should have an membership to the system and then would be able to request their choices on the internet. This is the main part of idea but it is more complex than it seems.

At first, system must determine the restaurant which will serve to customer whenever a new membership entry exists. Important point on determining restaurant is the address of customer. Most of restaurants serve to customers who live or to be served in the same county, not country or district. So, I would have taken county as the starting point to choose restaurant.

Secondly, every member of the system should have a unique member name and a password to confirm the membership.

Customers should change their own information in the system and choose a group of product as mentioned favourite choices so, they can select it when they come to system for shopping and go ahead fast.

The system does not accept a payment directly, only the instant payment type of customer should be taken on internet and solution of payment type must be provided while delivering process. Problems occuring via credit card usage on internet is exceeded by this method.

All product entries from the customer would be hold in a basket which is a database table containing some important information about customer and products that is choosen by customer to give it as an order.

4.1.2 Explanation of Restaurant Sale Management Application

This is the application for restaurants in chain. Whenever a customer gives an order, system will hold the needed information in database and use it for restaurants.

Every restaurant which has a membership to system will have a unique restaurant name a password to confirm it. Restaurants are not allowed to have a membership from the application.

In the sale management application, all orders whose the restaurant that is sign into its application, will be listed. Thus, restaurant will be able to manage orders whether they should be sent, they are delivered or returned.

This application has an obligation to enter the operator name to avoid not to be able to give responsibility to operators when a problem occurs. So, application must have some ability to add or remove operators for restaurant and there is.

Manager of the restaurant would add or remove operator using the name and SSN (Social Security Number) of the operator. A confirm part for password is included at this stage to avoid people that is not responsible make an action.

A change password part included also for managers.

System holds the closing time for every restaurant so, when it is the time, first application does not allow to customer whom the restaurant is closed. Because of a confirm mechanism that checks the closing times, every restaurant manager has an ability to enter the closing time.

4.2 Examining Requirements

It is very important to examining the requirements of the system. Both applications will use the same database and the database will be used by many restaurant at the same time. So, to have a simple database will slow down the application.

At this point, a server which has multi user support is needed. This server will contain a database which includes the needed tables. And I choose MS SQL Server 2000 to work.

The database would have different tables for every unique information used to differentiate entities. For example, an address information contains different parts like city, county, district... To avoid complexity and unproductive usage of the storage, different parts of address information must be hold in different tables.

4.3 Examining of Fast – Food Sale Application

Fast – Food Sale application is a web application because of it will be published on internet. Customers who visit the web site will see an home page including some advertisement of the company and a sign in hyperlink to the sign in page.

When they come to the sign in page they will see a form as below:

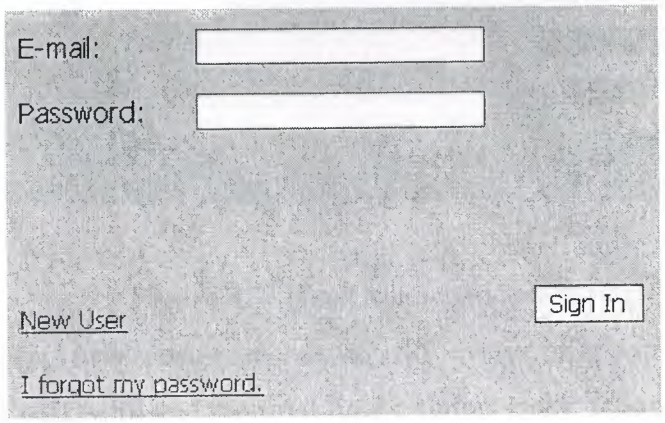


Figure 4.2: SignIn.aspx view

At this page, customer will enter its e-mail address and password to enter the system. E-mail addresses are unique for everyone so, I accept it as customer definer for both applications.

When you use ASP.NET to create a web application three important file to write code into are created. Extensions for the files are .html, .aspx, .aspx.cs. This files are used to creation of the web side of the page, user interface and functional code behind, respectively.

Whenever you add a component to the page, html code for it added to the file automatically. And we can write the code to code behind part by clicking on any component we want or form. Code behind part will automatically loaded with event function depending on component we click.

If customer is a new customer and haven't got any information in the system database, he/she must register. So, New User link must be clicked to enter the NewUser.aspx.

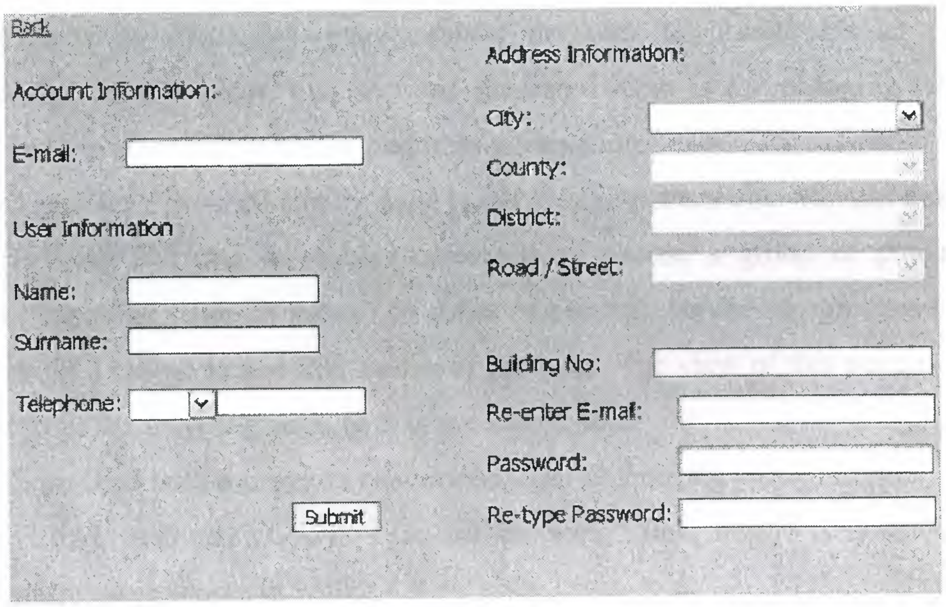


Figure 4.3: NewUser.aspx view

At this page, new customer has to fill every information, otherwise the validation controls will work and required field pointers make actions. A correct e-mail format validator is included also to check whether the entry is an e-mail or not.

At the page load County, District, Road/Street fields are disabled but in the case of choosing city and the other ones, every item will be enabled, respectively.

When a user signs in with his/her e-mail address and own password, Welcome page appears with the name of the restaurant.

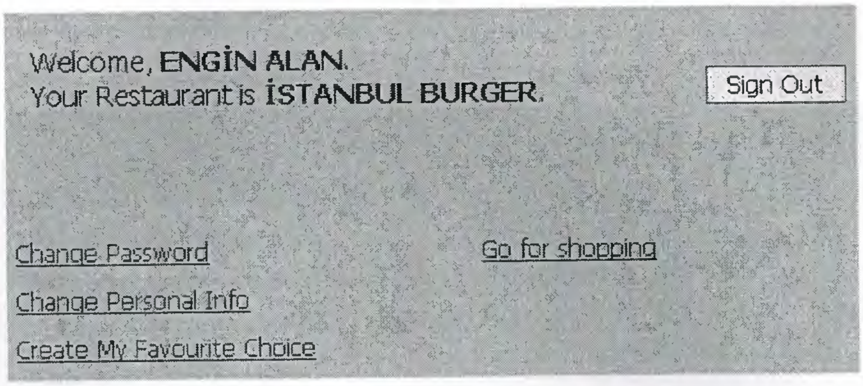

Welcome, ENGİN ALAN.
Your Restaurant is İSTANBUL BURGER.

Figure 4.4: Welcome.aspx view

This page also contains Change Password, Change Personal Info and Create My Favourite Choice links.

Change Password page enables the user to change his/her password to enter system. Old password is requested at this stage to confirm the user is our real customer.

Change Personal Info page enables the user to change his/her personal informations. At this point, two sections presented. One is for changing telephone number and the other is for changing address information.

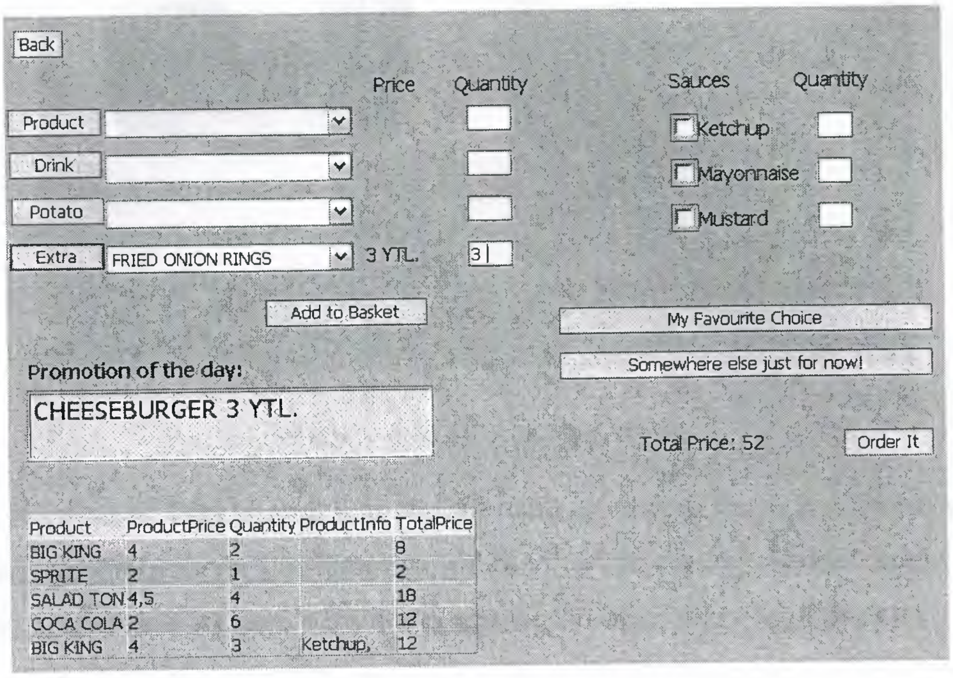
Create My Favourite Choice page is the most important one who has link at this stage I believe. Because, it enables customers to choose a group of product. So, whenever customer come to submit an order (shopping), he/she can go ahead fast by only clicking a button to add this choice of products. The view of this page is not too different from the shopping page, so it is not shown here.

"Sign Out" button provides customer a sign out from the system in safe.

When a customer click on "Go for shopping" link, he/she is directed to the shopping.aspx page shown in Figure 4.5.

In this page customer can choose any product which is chategorized into four groups and add it into the basket. When a new entry of product is added to the basket, it is also added to the list at the bottom of the page. This table is a view of the basket and includes every important information of the order. For every product quantity value must be entered between 1 and 20. And customer is provided to choose sauces and its quantity cannot exceed 3.

Page includes an pre-defined entry as "Promotion of the day". Every day one or more product is promoted by company so, in the case of choosing promoted product, customer is provided to have this product cheaper than other days.



Extra: FRIED ONION RINGS 3 YTL. 3

Promotion of the day:

CHEESEBURGER 3 YTL.

Total Price: 52

Product	ProductPrice	Quantity	ProductInfo	TotalPrice
BIG KING	4	2		8
SPRITE	2	1		2
SALAD TON	4,5	4		18
COCA COLA	2	6		12
BIG KING	4	3	Ketchup,	12

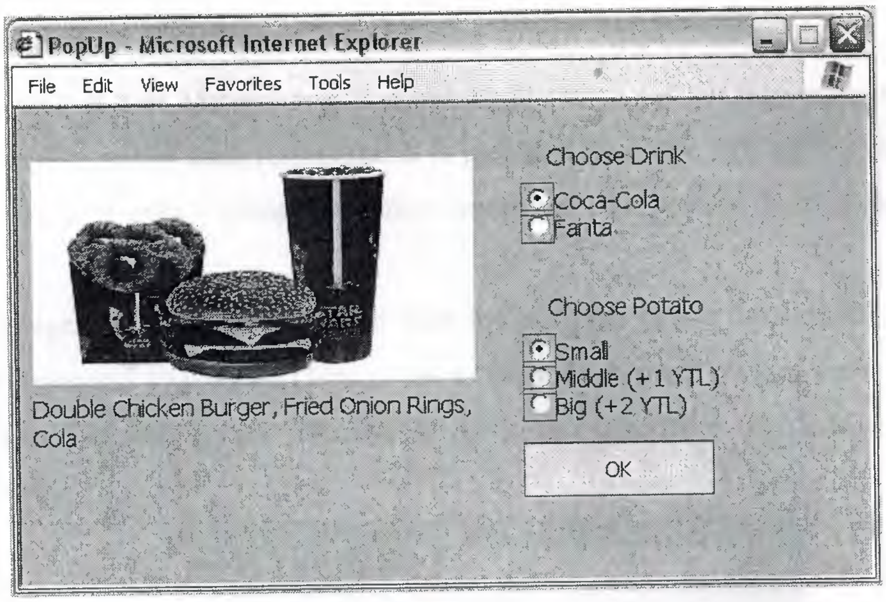
Figure 4.5: Welcome.aspx view

There is a button named as "Somewhere else just for now!" and it is used to send these products to another address different from one hold in database. In such a case, system choose a different restaurant if the given address is not served by the restaurant.

To send order to the system after choosing payment type, customer can click "Order It" button.

But before going to the payment page, I want to mention about the button next to drop down lists. Every category of product has a button named as category name. When a customer choose a product, enter its quantity and click the category-named button, a pop-up window appears with picture of the choosen product. Figure 4.6 is shown below.

In this page customer can see the picture of product and what product includes. Page also includes some extra choices for the customer depending of the product. For example when you choose a drink, patato choices do not appear in the pop-up. In this project extra choices for drink and potato have an static payment difference (couldn't be changed). For instance, when you middle potato 1 YTL difference you pay.


PopUp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
OK

Double Chicken Burger, Fried Onion Rings, Cola

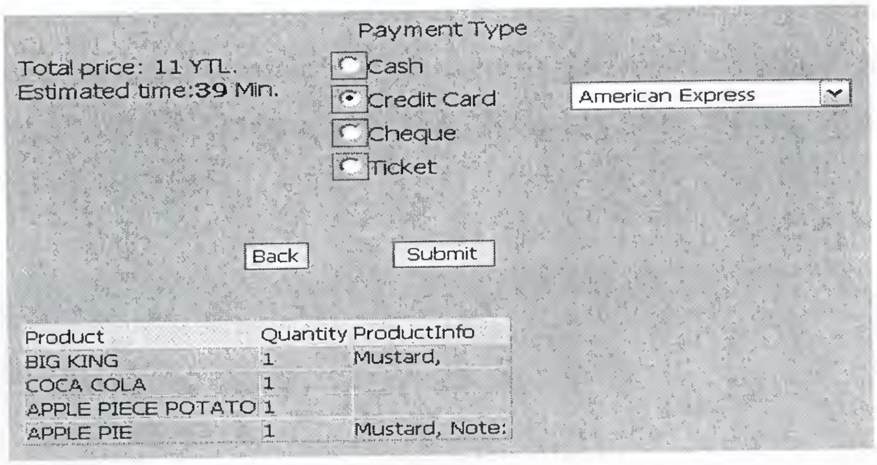
Choose Drink
[✓ Coca-Cola]
Fanta

Choose Potato
[✓ Small]
Middle (+1 YTL)
Big (+2 YTL)

Figure 4.6: PopUp.aspx view

Because of PopUp.aspx is a pop-up window, it appears in a different window and when customer hit the OK button, it is closed. While closing changes on the choices transmit to the shopping page via some components which every page contains.

When customer hit the "Order It" button on the shopping page, a new page come across the user with payment types. Figure 4.7 is shown below.



Total price: 11 YTL.
Estimated time:39 Min.

Payment Type
Cash
[✓ Credit Card] American Express
Cheque
Ticket

Product	Quantity	ProductInfo
BIG KING	1	Mustard,
COCA COLA	1	
APPLE PIECE POTATO	1	
APPLE PIE	1	Mustard, Note:

Figure 4.7: OrderIt.aspx view

In "OrderIt" page, customer can see every product in the basket. If he/she clicks on the "Back" button, shopping page appears again, but in such a case the basket of the customer is discharged and products are deleted.

At the bottom of the page total price and estimated time is shown. And at the right side of the page customer choose the payment type that he/she wants to use.

In the case of submit button is clicked a "registered successfully" page appears.

4.4 Examining of Restaurant Sale Management Application

Restaurant Sale Management is a web application used for managing orders that is in responsility of the each restaurant. It is prepared as a web application so, it is not needed to setup it onto a computer before using and can be used from anywhere via internet.

Manager of the restaurant must start the program (enter the system) using the restaurant name and password. There is not a "New Restaurant" part of the application, it is an action that only the administrator of the system can make. SignIn.aspx is shown as Figure 4.8.

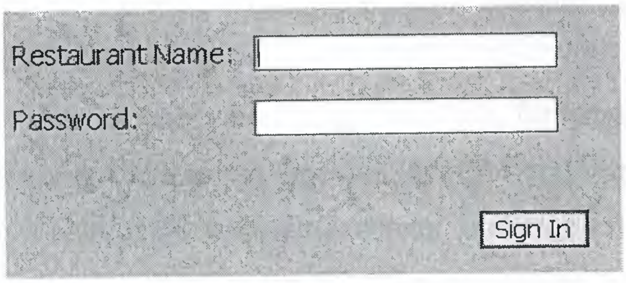


Figure 4.8: SignIn.aspx view

Page includes validations for required field (in this manner, both restaurant name and password) incorrect password. When there is not a problem about pasword or empty fields, if "Sign In" button is clicked, "Welcome" page appears. It is shown in Figure 4.9 below.

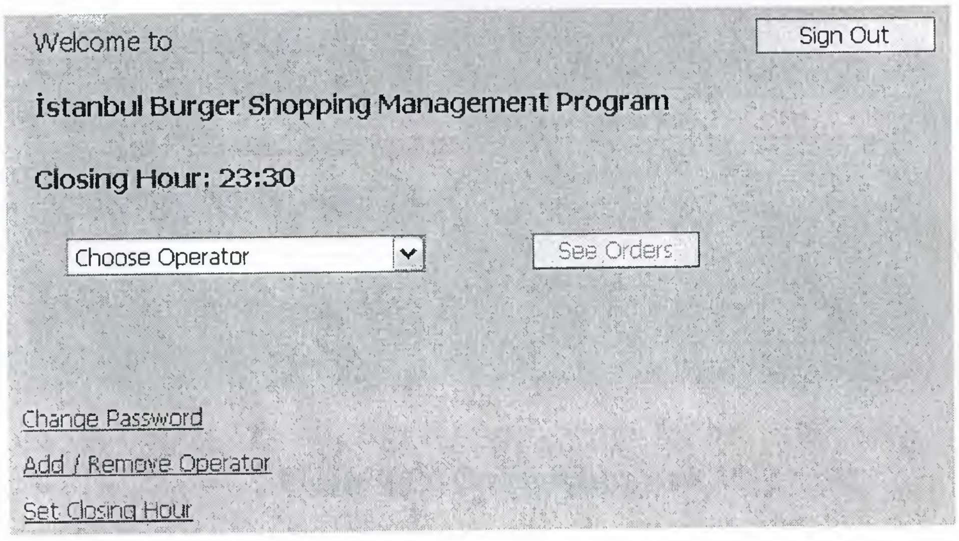


Figure 4.9: Welcome.aspx view

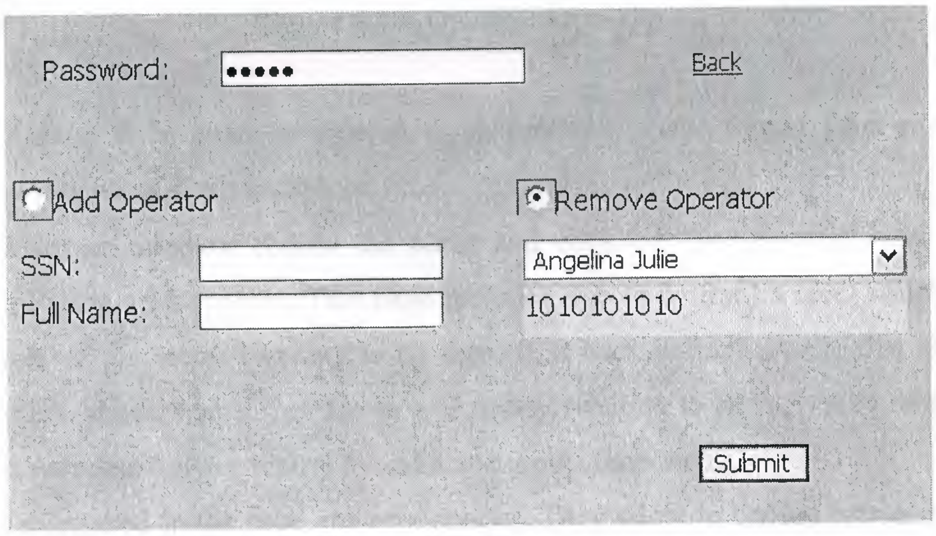
Page comes with the some information, eg. name and closing hour of the restaurant. A sign out button is there in the page to sign out in safe, too.

Firstly, I would like to mention about operator drop down list. Some restaurants have a potential to sell their products on the internet much more than other sale ways. For such restaurants, this application is getting more importance and in some cases only one copy of application may not be enough to meet orders. In such a case, more than one entry should be entered to the system and this means more than one computer and operator for a restaurant. So, to avoid confusion in the restaurant, operator choosing is an obligation to see and manage orders. Here, the confusion makes sense if a problem exist in the records of orders. In such a case, responsible operator can be determined taken an precaution easily.

Welcome page also includes there important link for administrator of the restaurant: Change Password, Add/Remove Operator, Set Closing Hour.

Change Password page enables the administrator to change restaurant password to enter system. Old password is requested at this stage to confirm the administrator is our real member.

Add/Remove Operator page provides the administrator to make action as adding or removing an operator using the operator name an SSN (Social Security Number) as shown in Figure 4.10.



Password: ●●●●● Back

Add Operator [✓ Remove Operator]

SSN:

Full Name:

Angelina Julie

1010101010

Submit

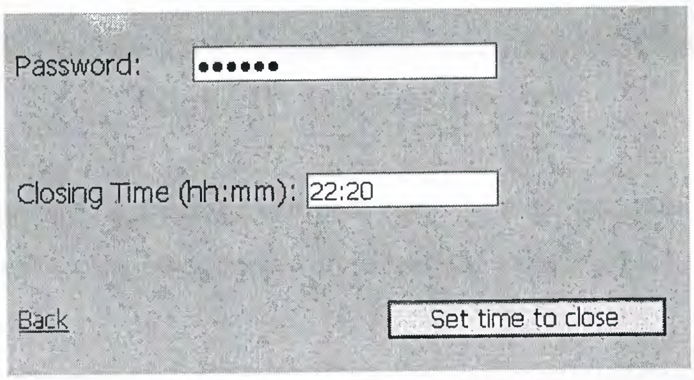
Figure 4.10: Operator.aspx view

To make an action as adding or removing operator, password entry is needed.

Administrator must choose operation type. If add operation is choosen SSN and full name of operator must be entered. If remove operation is choosen name of the operator must be selected from the drop down list. For both of the process, "Submit" button must be clicked.

In the case of adding a new operator, SSN is the primary key for operator so, it must be unique. Validation component of the page is registered to check whether there is an operator in database and is tried to enter the same one or not. So, it is impossible to add an existed operator to the system.

Set Closing Hour link enables the administrator to set the closing hour of the restaurant. If admin does not do an set operation, closing hour will stay the last set and the default value is 23:30.


Password: ••••••

Closing Time (hh:mm): 22:20

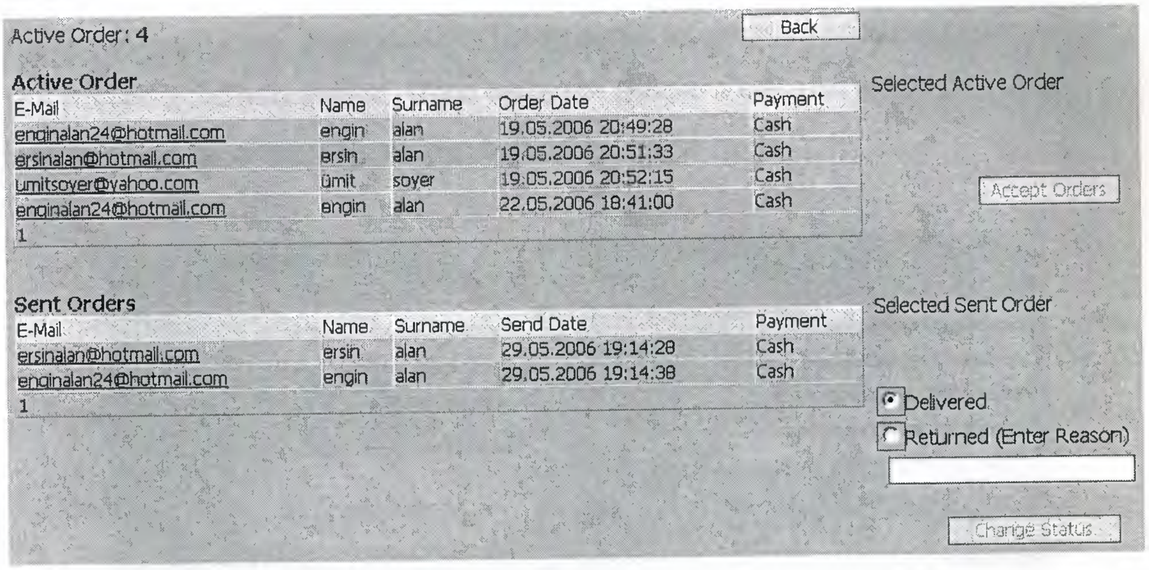
Figure 4.10: Operator.aspx view

Closing Time must be entered as determined in the format label otherwise, validation process will not accept the time.

When an operator choose the name and click on "See Orders" button, page shown in Figure 4.11 appears. This page includes (top to bottom) a label which shows the number of the active (waiting to be served), a back button, a table that show the active orders, another table that shows sent orders (waiting to be registered database as delivered succcess fully or returned) and some input components.

Tables used in the page are very special. They allow to choose entries by using the column of e-mail addresses. Selected Active Order and Selected Sent Order labels takes the name and surname of the customer when their e-mail address is choosen from the table.

Table information are grouped by order date ascending format so, shopping orders are listed as the time they come and served as FIFO logic.


Active Order: 4

Active Order

E-Mail	Name	Surname	Order Date	Payment
enginalan24@hotmail.com	engin	alan	19.05.2006 20:49:28	Cash
ersinalan@hotmail.com	ersin	alan	19.05.2006 20:51:33	Cash
umitsoyer@yahoo.com	ümit	soyer	19.05.2006 20:52:15	Cash
enginalan24@hotmail.com	engin	alan	22.05.2006 18:41:00	Cash
1				

Sent Orders

E-Mail	Name	Surname	Send Date	Payment
ersinalan@hotmail.com	ersin	alan	29.05.2006 19:14:28	Cash
enginalan24@hotmail.com	engin	alan	29.05.2006 19:14:38	Cash
1				

Selected Active Order

Selected Sent Order

[✓ Delivered] Returned (Enter Reason)

Figure 4.11: SeeOrders.aspx view

Any component on the page can post back the page automatically. This means that, any click event on any component will refresh the page so, it is not needed to make another action.

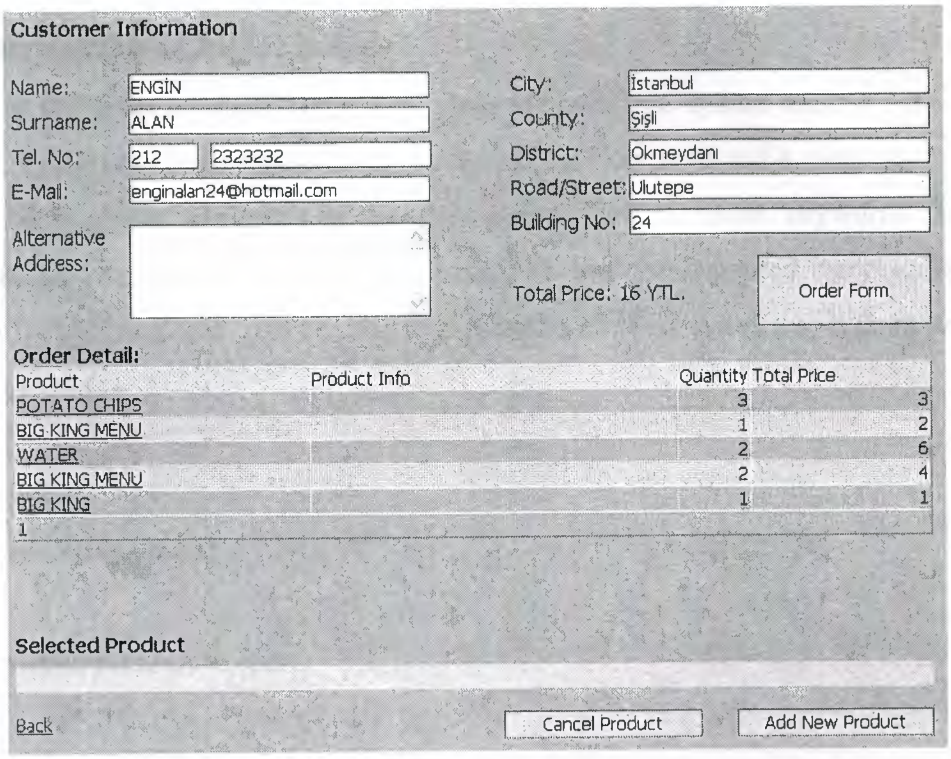
Another very important attribute of the page is the validation message of "There are still customers to be served!". If some customers send any orders before closing time of the restaurant, they should be served. Hence, a validation is created to check whether waiting customer(s) existed or not. If exists, validation does not allow the operator sign out so, either every customer should be called and orders should be canceled one by one (on the AcceptOrder.aspx) or every order should be delivered.

Both of the lists have page numbers after the last row of tables. This page numbers are very important being as the visual part of application. To provide a easy-to-use page operators, every component on the page fits into the page. So, tables can carry only five of first rows in order. Next "fives" take a part in different pages. And every table carries the page number which it has. Operator doesn't care the page numbers, because every entry takes place in first page as accepting the orders.

When operator choose a name from the sent orders list, choosen order can be registered to database delivered by clicking "Change Status" button. If order is returned, reason of the return must be entered and same button should be clicked again.

If operator want to accept an order, he/she should choose it from active orders list and "Accept Order" should be clicked. The page "AcceptOrders.aspx" will appears in the case of action.

"AcceptOrder" page includes the personal information of the selected customer and his/her order information. If the customer choose a different address to send order, it is shown in the alternative address box.



Customer Information

Name:	ENGİN	City:	İstanbul
Surname:	ALAN	County:	Şişli
Tel. No:	212 2323232	District:	Okmeydanı
E-Mail:	enginalan24@hotmail.com	Road/Street:	Ulutepe
Alternative Address:		Building No:	24
		Total Price:	16 YTL.

Order Detail:

Product	Product Info	Quantity	Total Price
POTATO CHIPS		3	3
BIG KING MENU		1	2
WATER		2	6
BIG KING MENU		2	4
BIG KING		1	1

1

Selected Product

Figure 4.11: SeeOrders.aspx view

In this page, operator can choose any product and cancel it or add a new product. It may seem meanless but, in the case of any product is not in the stock, customer should be called and took an opinion to operate on the order. Order may be cancelled by canceling every item in list. "Selected Product" label shows the name of the selected product.

If the order is registered to database by clicking "Order Form" button, it takes place in the See Order page as a sent order and waits answer from the customer to be registered database as delivered or returned.

I will not mention about "Add New Product" page, because it is very similar to the shopping and my favourite choice pages in first application.

CHAPTER FIVE: CODE STRUCTURES AND PROGRAMMING TECHNICS USED IN APPLICATIONS

In this chapter, general code structure of C# will be explained using code examples from project. The main reason of writing these explanation to help students who think to prepare a web application in ASP.NET as graduation project.

5.1 User Defined Classes in C#

In C#, most of object definition, class methods and syntax has been taken from C and C++. So, to create a class in C# we use "class" keyword and a class name. Public, private or protected attributes of the class determined via these keywords. To write methods for variable's get and set operations are implemented very easily. The keywords "get" and "set" can be used to create this method. An example from procejt is shown below:

```
using System;
namespace Graduation2
{
        public class Customer
        {
                private string sUserEMail;
                private int iRestaurantID;
                public string UserEMail
                {
                        get{return sUserEMail;}
                        set{sUserEMail = value;}
                }
                public Customer(){}
        }
}
```

In the example code above, "using" keyword is used as "include" in C. System header file is the main one for creating a class. The keyword "namespace" is used to determine the solution name we use class in. The method Customer() is the constructor of the class. It is called automatically when a component from this class is created.

5.2 How to Validate a Control

Visual Studio .NET comes with built-in validation controls for use in ASPX pages. When you validate a control, you're checking whether a control has data in it and conforms to a specific pattern, (such as an email address), or you're checking the range of data that has been entered.

In ASP, you either had to process the validation on the server and send the page back to the browser if data wasn't correctly entered, or you had to write complex JavaScript to check the validity of control data. ASP.NET has five built-in validation controls in the Toolbox that you can simply drag to a form and associate with a control.

RequiredFieldValidator: Forces the user to enter value into the specified control.

CompareValidator: Compares a user's entry against a constant value, or against a property value of another control, using a comparison operator.

RangeValidator: Checks that a user's entry is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

RegularExpressionValidator: Checks that the entry matches a pattern defined by a regular expression.

CustomValidator: Checks the user's entry using validation logic that you write yourself. This type of validation allows you to check for values derived at runtime.

Each validation control has properties specific to the functionality the control provides. For example, the RequiredFieldValidator has a ControlToValidate property and an ErrorMessage property. The ControlToValidate property takes the ID of a valid control on the form. The RegularExpressionValidator uses the regular expression syntax of .NET to validate the data entered in a control against a regular expression pattern.

Each validation control has a Text property and an ErrorMessage property. The Text property is like the Text property of a Label control - it simply displays text. This could be used to display Required or an * for a RequiredFieldValidator control. The ErrorMessage property displays if an error occurs in the validation.

A custom validator example from project is shown below:

```
private void CustomValidator1_ServerValidate
(object source, System.Web.UI.WebControls.ServerValidateEventArgs args)
{
        if (IsThereRestaurant(txtName.Text, txtPassword.Text))
                Response.Redirect("Welcome.aspx");
        else
                args.IsValid =false;
}
```

In the example, there is a user-defined method as:

bool IsThereRestaurant (string a, string b)

This method has a return type of boolean. This method takes name and password, if there is a corresponding user in database return a true, if not returns false. So, in the case of returning false, "args", the parameter of the custom validator, will be invalid (using IsValid attribute). That is, custom validator sends to customer an error message that is predefined.

5.3 Managing State in ASP.NET Web Applications

The Internet is a stateless application. That means every call to the server is separate from the next or the previous call. Unlike Windows Forms applications in which you can set global variables and have a bunch of hidden forms open to maintain the state of the data, the Web doesn't afford you that luxury. For internet applications this is done by maintaining state.

State can be maintained in several ways, and some ways are better than others. Where you maintain state can be broken down into two categories: on the client and on the server.

Session and Application are "on server state" and I only use session and application states in my project. Because it is enouh to save any data and share between the pages so, I will explain only the session application.

Using Session State: Using session-level state enables you to track variable data for a user throughout his visit to your Web site. Using the HttpSessionState class, you can use the same key-value pair syntax for Session objects that you use for ViewState. Each time a browser hits your site, IIS creates a unique session ID for the browser session if you access the Session object in code. If you don't reference the Session object in code, IIS doesn't create a unique ID for the end user's session on your site.

Session information is useful to save data between multiple page calls. For example, when I visit my bank's Web site, I must enter my username and password every time I visit, which means the bank isn't storing such high-security data in a cookie. But after I log in to the site, each page I navigate knows who I am, so the data is being passed to all pages for my session through the use of session state. To set or retrieve values for session state, you use this syntax:

```
If Session["UserID"] = "Engin"
Response.Write(Session["UserID"]);
```

After the browser window is closed or the end user navigates to another site, the session data is lost. When the end user returns to the site, you must create new session information based on the user's current session.

Using Application State: Application-level state is the top level in the state management hierarchy. The first time someone accesses your Web site, the Application object for the site is created.

The Application object is alive until the server is rebooted, IIS is restarted, or a new copy of the Web site is deployed. In the Global.asax file, there are Application_OnStart and Application_OnEnd events that you can write code to respond to; they're similar to the Session_OnStart and Session_OnEnd events.

The difference is that application-level variables are global for all sessions for your Web site, not for individual users.

5.4 Errors and Exceptions

If you want to manage exception handling:

- Understand why errors occur in applications.
- Learn about structured exception handling in .NET and how the common language runtime implements exceptions.
- Use Try, Catch, and Finally blocks to avoid runtime errors.
- Learn how to use all the debugging features available in Visual Studio .NET for ASP.NET applications.

5.4.1 Why Errors Happen

In the course of writing any type of computer program, three types of errors can occur:

- Syntax errors
- Logic errors
- Runtime errors

How to Avoid Syntax Errors: Syntax errors occur when you misspell a variable name, object name, or reference an object incorrectly. Syntax errors are the easiest errors to avoid, because Visual Studio .NET lets you know if a keyword or variable is misspelled. You're notified of syntax errors in both C# and Visual Basic .NET by squiggly lines that appear under code that isn't correct after you type a line of code. When a squiggly line appears under your code, you can hover your mouse over the squiggly line and a tooltip lets you know what error is occurring.

How to Avoid Logic Errors: Logic errors are the hardest errors to avoid. A logic error is a flaw in the way you designed a function, or group of functions, to perform a specific task. Logic errors aren't normally discovered until an application is out in production and being used by end users. This types of error is unavoidable. The only real way to minimize logic errors is to write your applications based on specifications that have been well thought out and clearly defined.

How to Avoid Runtime Errors: Runtime errors are errors that occur at runtime, but aren't handled by an error handler in your code. Runtime errors are the worst type of error that can occur because they directly affect the end user of your application. Runtime errors can also be called laziness errors, because they can be completely avoided simply by implementing some sort of error handling in your code. When an unexpected error occurs and there's no error-handling routine in your application, the application notifies the end user of the error with a very offensive message and crashes immediately.

5.4.2 Understanding Exceptions in .NET

To make your code compliant with the Common Language Specification (CLS), you must use the exception object from the System.Exception class in the Framework Class Library (FCL) in C#.

The Exception object contains the information you need about the last error that occurred in your application. When an error occurs, an Exception object is created. So, when a block of code causes an error, an exception is thrown and the first block of code in the stack that has an exception handler takes care of that exception. If there's no exception-handling code, a runtime error occurs, and your application terminates.

5.4.3 Handling Exceptions with Try/Catch/Finally

The syntax for Try/Catch/Finally blocks can be explained with this rule:

- In C#, each try statement must end with a curly brace, and include at least one catch statement or one finally statement.

Algorithm in one sentence to handle an exception is that you're trying to execute some code, catching an error if one occurs, and finally doing something when the code succeeds or fails. The following C# syntax will give you an idea of what a Try/Catch/Finally block might look like:

C# Code structure:

```
try
    {
            // This is code you are attempting to execute
    }
catch
    {
            // This is code that executes if an exception occurs
    }
finally
    {
            // This is code that runs after the Try or Catch statements
    }
```

An example from the project:

```
try
{
        string x = SystemDate();
        string sql = "INSERT INTO Users Values(...)";
        con.Open();
        SqlCommand da = new SqlCommand(sql,con);
        da.ExecuteNonQuery();
        Response.Redirect("Saved.aspx");
        con.Dispose();
}
```

```
catch(SqlException exp)
{
        exc = exp.Message;
        Session["errormessage"]= exc ;
        Response.Redirect("Error.aspx");
}
```

In this example an sql inserting database operation is made. If server is not opened or if a problem exists about connection, error message will be sent to another page and it is shown there.

Understanding the Catch Statement: The Catch statement filters exceptions based on the code executed in the Try statement. There are a few ways to handle Catch statements:

- Catch a generic error
- Catch a specific error, using Catch As, based on the type of exception that occurred

In C#, you can specify a specific error in parentheses immediately following the catch statement. Catch specific errors using Catch As or Catch When and then Catch a fallback generic error.

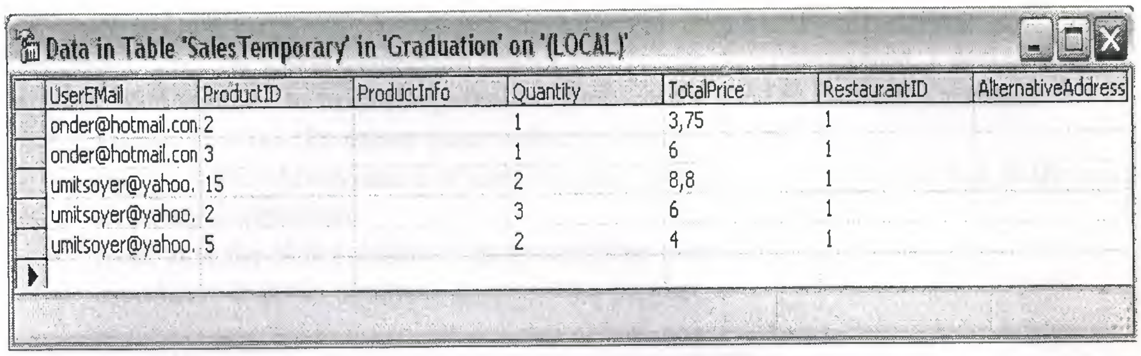
In C#, you get a compiler error if you attempt to catch a specific exception after having specified a generic Catch statement.

5.5 Connecting to the Database:

There are several ways to setup a connection to database but, I used the most effective one in my graduation project. To show how to connect a database and transfer data to a grid control, I will explain every step by using example codes from the project.

For example, I have used a grid component to show customer what he/she has choosen as products in the first application. All the steps are presented below:

As the first action of explanation, we need a database table to transfer data into grid. I created a table named as “SalesTemporary” which is used as a holder for the products in basket.

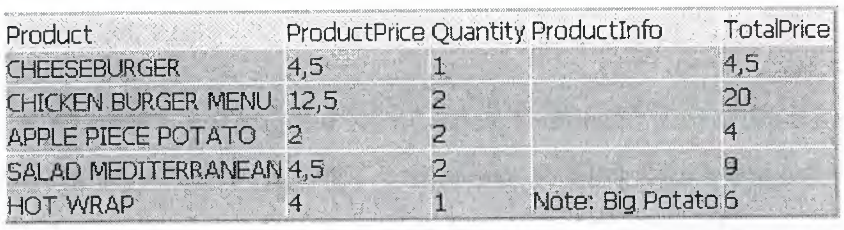


Data in Table 'SalesTemporary' in 'Graduation' on '(LOCAL)'

UserEMail	ProductID	ProductInfo	Quantity	TotalPrice	RestaurantID	AlternativeAddress
onder@hotmail.con	2		1	3,75	1	
onder@hotmail.con	3		1	6	1	
umitsoyer@yahoo.	15		2	8,8	1	
umitsoyer@yahoo.	2		3	6	1	
umitsoyer@yahoo.	5		2	4	1	

Figure 5.1: SalesTemporary table in database

After creating table, we need to add a grid component to show data on. Grid component is a html table to show and edit data on and it is the easiest to use and complex data component in .NET platform.



Product	ProductPrice	Quantity	ProductInfo	TotalPrice
CHEESEBURGER	4,5	1		4,5
CHICKEN BURGER MENU	12,5	2		20
APPLE PIECE POTATO	2	2		4
SALAD MEDITERRANEAN	4,5	2		9
HOT WRAP	4	1	Note: Big Potato	6

Figure 5.2: Products in Basket grid

This is the grid which is connected to database so, it is full of data shown. Now, we will see the code behind part of project and investigate code lines to have this full of data grid after adding it onto form by drag and dropping.

- using System.Data.SqlClient;

This is the first code line we write. It transfers header file to use data components in our project. So, we can use SqlConnection, SqlDataAdapter and DataSet components.

SqlConnection is the one of database connection in ASP.NET and used for connection to SQL Server database. The other object is the OleDbconnection which is used for connecting other databases such as Access, Oracle but, I will not mention about it because, I have not needed to use.

```
SqlConnection con;
string connection = "server=localhost;uid=sa;pwd=;database=Graduation";
con = new SqlConnection(connection);
```

This code lines may seem you meanless at first but I will explain everything. The object "con" is a SqlConnection object that is used for connection. I define a string variable "connection" to hold the parameters which are used to connect database.

- *server:* It takes the server name which is localhost here.
- *uid:* It is the abbraviation of user id, and "sa", the default user in sql, is the user id of my connection.
- *pwd:* It is the abbraviation of password that user of the system has.
- *database:* It is the database name of the project.

So, con object is constructed with the parameter holder string and we can use it on application.

```
string GetOrder = "SELECT Product.Product, Product.ProductPrice, " +
                  "SalesTemporary.Quantity, SalesTemporary.ProductInfo, "+
                  "SalesTemporary.TotalPrice FROM SalesTemporary " +
                  "Product ON SalesTemporary.ProductID = Product.ProductID " +
                  "WHERE (SalesTemporary.UserEMail = '" + email + "') ";
SqlDataAdapter da = new SqlDataAdapter(GetOrder,con);
DataSet ds = new DataSet();
da.Fill(ds);
DataGrid1.DataSource = ds;
DataGrid1.DataBind();
```

Here, GetOrder is holding the select string to transfer data from database table. SqlDataAdapter is very useful Asp.NET component to filter data using sql-select string and a connection. DataSet is used to hold data filtered by the data adapter component.

At the end, DataSource method of DataGrid object is used to point data to grid component but it just points. To transfer data columns into grid component, we use DataBind method with no parameter after pointing process.

5.6 Problems I Faced on Project and Solutions

Authentication Mode Error: Whether you write a code or not, when you create an web application on ASP.NET and debug it, if you face with an error about Authentication Mode=Windows and Server "/" Application error, you should apply this steps:

- Open the Control Panel
- Find Administrative Tools and open it.
- Double click on the IIS icon.
- Right-click on your application directory and click properties
- Click on the create button to creat a virtual directory for IIS.

Integrated Windows Authentication Is Not Enabled: If you use a connection to SQL database and your connection properties is choosen as windows autentication, you must construct your project as integrated windows autentication.

- Open the Control Panel
- Find Administrative Tools and open it.
- Right-click on Web Sites directory and click properties
- From the Directory security tab, click edit button.
- Choose Integrated Windows Security option click Apply
- Choose every option in coming window.

Not Viewing System Objects in SQL Server: To show only the tables and objects you create, not the system has

- Right-click on LOCAL
- Click SQL Server Registration Properties.
- Uncheck the Show System Databases and Objects.

Drop Down List Not Working: If you add some lines of code to Selected_Index_Changed method of the drop down list component, you are probably using a server control and it is needed to post back the page to make an action to the server and refresh page. Solution is to changing AutoPostBack property of drop down list component to true value. It is a common solution for other type of input components.

Problem About Sending E-Mail: You need to have a virtual mail server to send mails in ASP.NET, so you should

- Open the Control Panel
- Find Administrative Tools and open it.
- Double click on the IIS icon.
- Right-click on the Default SMTP Virtual Server and click properties
- On the Access tab, you should click Relay button
- On the page coming, click Add button and enter the value "127.0.0.1" which is your local IP address.
- Click OK button to finish work.

ASP.NET Not Working (Opening Problem): If you cannot open ASP.NET solutions or not able to create a new solution, check the Inetpub\WWWroot directory whether you have "asp_client_script" directory or not. If you haven't got it, you should

- From Start \ Programs \ Microsoft Visual Studio .Net 2003 \ Visual Studio Tools, run the "Visual Studio .NET 2003 Command Prompt"
- Enter the command "aspnet_regiss -i" and hit the enter key.

This command will copy the needed files to your computer.

Validation Problem of Text Box: When you add some text boxes to your web page and relate them with validation controls, it is wished that validation will work when the page is posted back. But, if the validation control works without post back, just with tab steps on components, you should change the EnableClientScript property of text box (another input components in different cases) to false.

CONCLUSION

Web applications are not the newest idea in programming but, their importance is getting increased day by day with new technology of ASP.NET. Very useful features of .NET platform and development of coding in secure for applications can be observed at first glance.

Although most of programming structure and techniques haven't been able to use in, applications working on the internet are the most popular way to develop software for companies. Because, when you develope a web application, you do not need to setup it on computers to work with. Just putting the needed files on server and publishing it to internet, you can reach your program whereever you are.

I think this project can be used as a real world application by companies. Most of ideas used in application are examined by investigating working principles for sale on internet of big companies such as Mc Donalds and Burger King.

Becase of the importance of development in web based applications, I decided to prepare my project as such an application. I got lots of experience about subject through developing the project. It is not the first experiment for me, but is the most important one. I feel proud of seeing my project to work efficiently.

I would like to add new technologies into my project and develope it in future, may be it is the best way to get a big step into my job.

REFERENCES

[1] Walther S., ASP.NET Kick Start, Sams Publishing, December 16, 2002.

[2] Onion F., Essential ASP.NET with Examples in C#, Addison Wesley, February 11, 2003.

[3] Beres J., Sams Teach Yourself Visual Studio .NET 2003 in 21 Days, Sams Publishing, January 14, 2003.

[4] Kennedy B., Musciano C., HTML & XHTML: The Definitive Guide, 5th Edition, O'Reilly Publishing, August 2002.

[6] Watkins D., Hammond M., Abrams B., Programming in the .NET Environment, Addison Wesley Publishing, November 06, 2002.

[7] Terence J.J., Payet R.N., ADO .NET Programming, Wordware Publishing, 2003.

[8] Shepherd G., "DataList vs. DataGrid in ASP.NET", "http://msdn.microsoft.com/msdnmag/issues/01/12/asp/"

[9] Salman A., "DataSet Made Simple", "http://www.csharpfriends.com/Articles/getArticle.aspx?articleID=2"

[10] Mitchell S., "Deciding When to Use the DataGrid, DataList or Repeater", 2003.

[11] GotDotNet Message Boards, SQL Server Failure, "http://www.gotdotnet.com/Community/MessageBoard/Thread.aspx?id=270585" Retrieved: 10/11/2004

[12] Manoj R., "Session Management", Retrieved: 03 Mar, 2004 "http://www.dotnetspider.com/kb/Article118.aspx"