Near East University

Faculty of Engineering

Department of Computer Engineering

Medical Consultancy with ASP

Graduation Project
COM-400

Student:	Mert KIRAN
Supervisor:	Ümit İLHAN

Nicosia-2003

ACKNOWLEDGMENT

At first I want to thank to God, creator of everything, for give me chance and opportunity for studying. I am going to be engineer because of my family's supports. I am saying 'thank you to God' again because of my mother and father. They give me second chance for studying and I think I used this chance very well. Special thanks to my grandmother, here I always feel her prays. Thanks to my all relatives. And again thanks to God for my special person, who support me from 6 years for what I did. Thank you Esra. Thanks to my brothers Osman and Antonyo for giving me advice to choose Near East University. Thanks to Mr Suat Günsel for giving me and my many friends scholarship. In our department, I gratitude everybody from cleaner up to dean of the faculty. Special thanks to Mr Tayseer Al-Shanableh for me advising. One special thanks to Mr Ümit İlhan who is the most important computer lectures' lecturer. Not only in this project, but also in my another computer problems he always helped me.

Finally I want to thank my all friends not only Turkish ,but also Arabic countries origin friends. I learned from meeting them the most important thing in the world is being human. Different nationality, cultures and languages are not important for communicating people. The main thing is being hearts together.

ABSTRACT

Health is the most important thing in the human life. In today's world people can earn money, they can lose money also. When people lose money they can earn again. But for human health this is not completely true. If the disease is serious, may be you cant take your health back. When you lose your health you cant be as healthier times.

By developing technology, human's time takes more important because of sparing time for sufficient activities. People want to take information quickly. Internet makes our life better. We can take any information about something in a few seconds.

In doktorkibris.com I want to give information to people about health and I prepared very complicated database about medical consultancy and all doctors' addresses and their information. There are search pages for finding any doctor or dentists and also pharmacies. People can find any of them easily.

I used ASP which helps me to publish database in internet. I give detailed information about ASP and it's development periods and then I give detailed information about ASP's objects and functions.

Finally in my graduation project I tried to prepare an efficient thing, and now I can see that I did what I wanted to do.

CONTENTS

INTRODUCTION

Today technology is developing not every day. Technology is developing in every seconds. Because in all over the world, millions of people are trying to develop themselves, and everybody wants to use technology for themselves. About from 5 years, every month i buy a magazine about computer and i collect all of them. Sometimes I checked old magazines, and at this time I can see technologies' developing. For example a video card's advertisement says that "A force like no other". But it is very enjoyable for nowadays. This video card is just 4 MB. It means that today you cant use this on a lot of Standard graphic programs. Nowadays for new video games requires minimum 16 or 32 MB video cards. From this we can see how technology develops.

Internet also develops with technology. I can remember that in 1996 when i start to use internet, in altavista search engine, i was searched Turkey, and it found just about 3000 pages. But today when i search Turkey, it finds millions of pages. At the begining people use internet for sharing information, documents and communication. But when internet is being to Standard for a lot of people, companies start to interest internet Technologies. And in big companies they made a new department for Technologies. They want to use internet for customer relationships, collecting interests of their customers. They do this with databases. Database is very important for people.

From primary school we use databases although we dont know where we use. In my opinion all history is databases. All mathematics databases. Geography also. Because all of them prepared by a lot of different people.

In this point with my Project i will show you how database prepared and used for internet. We cant use directly our database from the internet. For publishing database or searching something in our database we must use some languages. There are many languages for using internet, such as perl, cgi, php and asp. Last year when i was taking com 340 course, I met with ASP. And after this course, I interested with this too much. So in my graduation Project i wanted to make something with ASP.

With asp what can I do. I searched and i want to make efficiency things. After I graduate, if people use my Project i will be glad. So i decided to make Project with Mr Ümit İlhan's advices, medical database for Turkish Republic of North Cyprus. It's name is doktorkibris.com.

For this Project i read hundreds of pages from internet, 3 books, i spare hours of time in internet and prepared hundreds of pages. Just collecting and preparing my databases took may 1 week. But after finish it i can see that it is really what i want.

CHAPTER 1: BEFORE STARTING ASP, MAIN QUESTIONS ABOUT IT.

At the beginning of in my work, I searched a lot of sources about ASP. I collected the main topics of ASP as questions and answers. I can say that these are Frequently Asked Questions (FAQ) for ASP.

Let's start from the main question:

What is Active Server Pages?

Microsoft® Active Server Pages (ASP) is the server-side execution environment in Microsoft Internet Information Server (IIS) 3.0 that enables you to run ActiveX™ scripts and ActiveX server components on the server. By combining scripts and components, developers can create dynamic content and powerful Web-based applications easily.

What is dynamic content?

Web pages that are customized for each user on the fly, based upon their actions or requests. For example, new visitors to your site can be shown a different welcome page than returning users see, or pages in an online catalog can be queries to a database so customers always see the most current information and availability.

Who should use Active Server Pages?

Organizations will use the Active Server Pages technology to put a Web front end on existing business solutions, or to create entirely new Web-based applications. Since ASP provides a very open development environment, with support for both Microsoft Visual Basic®, Scripting Edition (VBScript) and JScript™, organizations can leverage the investments they already have in these scripting languages.

Who can create dynamic content with Active Server Pages?

Webmasters, information systems (IS) professionals, and programmers familiar with Hypertext Markup Language (HTML) and languages such as Microsoft Visual Basic, JavaScript, PERL, REXX, or C++.

What do I need to run Active Server Pages?

The Active Server Pages feature of IIS 3.0 requires Microsoft Windows NT® Server 4.0 running IIS 2.0 or Windows NT Workstation 4.0 running Peer Web Services.

The ASP feature does not require Service Pack 2 for Windows NT 4.0, although it is recommended that all Windows NT systems receive the Service Pack 2 updates. The ASP feature contains a subset of the updates found in Service Pack 2.

Installation of ASP will upgrade IIS version 2.0 to version 3.0. The other IIS 3.0 features—Index Server 1.1, Microsoft NetShow™, FrontPage® 97 Server Extensions, and Crystal Reports—add significant functionality to IIS, but are not required to take advantage of ASP.

How much will Active Server Pages cost?

Active Server Pages is a component of IIS 3.0, which is a free, downloadable, and integrated feature of Windows NT Server 4.0.

Features

What can Active Server Pages do for my business?

Active Server Pages can develop a new generation of Web-based applications, including extending sales and customer service to the Web, and providing access to corporate databases and applications to any browser on an intranet.

What ActiveX Server Components are supported?

Active Server Pages allows organizations to extend the power of scripting on the server with ActiveX server components. These components can be created using Microsoft Visual Basic, Visual C++®, Java, and other languages.

What scripting languages does Active Server Pages support?

Active Server Pages provides native support for both Microsoft JScript and VBScript. ActiveX scripting plug-ins are available for REXX, PERL, and Python.

What browsers does Active Server Pages support?

Active Server Pages can work with any Web browser. The output of an ASP file is plain HTML, the content of which can be customized for the capabilities of the client.

Does Active Server Pages maintain state for me?

Yes. Active Server Pages allows you to define application and session variables that can be carried across multiple pages in a Web site. This can be as simple as remembering a user's name, and it is necessary in applications such as online shopping to track product selections.

What about legacy data?

Active Server Pages makes it easy to bring legacy database applications to the Web.

Implementation Questions

On which platforms does Active Server Pages run?

Active Server Pages will run on Microsoft Windows NT Server 4.0, Windows NT Workstation 4.0 with Peer Web Services, and Microsoft Windows® 95 with Personal Web Server. Windows NT 3.51 is NOT supported. Windows NT 4.0 on MIPS is also not supported by ASP.

Is Active Server Pages secure?

Yes. Active Server Pages is a component of Internet Information Server, and thus uses Windows NT Security. ASP files can be easily restricted to just certain users through secure Windows NT authentication, basic Web authentication, or client-side certificates. For additional security, all client-to-server communications can be secured with Secure Sockets Layer (SSL).

What data sources can my Web application integrate with?

An Active Server Pages application can integrate with any ODBC-compliant databases including Microsoft SQL Server™, Oracle, Sybase, Informix, and DB2 databases. Any OLE 2 application, such as Lotus Notes or Microsoft Excel and as in my project Microsoft

Access can also be scripted to access or process information. You can also write components to access online data feeds and legacy mainframes.

Can Active Server Pages use Java?

Yes. Active Server Pages supports ActiveX server components written in any language, including Java. In addition, ASP includes the Microsoft Windows reference standard Java Virtual Machine.

What does Active Server Pages do better than other Web application tools?

Active Server Pages allows you to quickly bring your existing skills and knowledge, data sources, components, and applications to the Web. Other tools create either static HTML or lock you into a non-standard programming model or language. ASP is based upon the leading industry standards, making it easy to build, maintain, and evolve powerful interactive Web applications.

CHAPTER 2: OBJECTS OF ASP

The Active Server Pages (ASP) framework provides six built-in objects:

- **Application**
- **ObjectContext**
- **Request**
- **Response**
- **Server**
- **Session**

Application Object

You can use the **Application** object to share information among all users of a given application. An ASP-based application is defined as all the .asp files in a virtual directory and its subdirectories. Because the **Application** object can be shared by more than one user, there are **Lock** and **Unlock** methods to ensure that multiple users do not try to alter a property simultaneously.

Collections

Contents	Contains all of the items that have been added to the Application through script commands.
StaticObjects	Contains all of the objects added to the session with the <OBJECT> tag.

Methods

Lock	The **Lock** method prevents other clients from modifying **Application** object properties.
Unlock	The **Unlock** method allows other clients to modify **Application** object properties.

Events

Application_OnEnd

Application_OnStart

Scripts for the preceding events are declared in the global.asa file.

Application_OnEnd

The **Application_OnEnd** event occurs when the application quits, after the **Session_OnEnd** event. Only the **Application** and **Server** built-in objects are available.

Syntax

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>

Sub Application_OnEnd

. . .

End Sub

</SCRIPT>
```

Parameters

ScriptLanguage

Specifies the scripting language used to write the event script. It may be any supported scripting language, such as VBScript or JScript. If more than one event uses the same scripting language, they can be combined under a single set of <SCRIPT> tags.

Remarks

You cannot call the **MapPath** method in the **Application_OnEnd** script.

Application_OnStart

The **Application_OnStart** event occurs before the first new session is created, that is, before the **Session_OnStart** event. Only the **Application** and **Server** built-in objects are available. Referencing the **Session**, **Request**, or **Response** objects in the **Application_OnStart** event script causes an error.

Syntax

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
Sub Application_OnStart
. . .
End Sub
</SCRIPT>
```

Parameters

ScriptLanguage

Specifies the scripting language used to write the event script. It may be any supported scripting language, such as VB Script or JScript. If more than one event uses the same scripting language, they can be combined under a single set of <SCRIPT> tags.

What is Global.asa

The Global.asa file is an optional file in which you can specify event scripts and declare objects that have session or application scope. It is not a content file displayed to the users; instead it stores event information and objects used globally by the application. This file must be named Global.asa and must be stored in the root directory of the application. An application can only have one Global.asa file.

Global.asa files can contain only the following:

- **Application events**
- **Session events**
- **<OBJECT> Declarations**
- **TypeLibrary Declarations**

If you include script that is not enclosed by <SCRIPT> tags, or that defines an object that does not have session or application scope, the server returns an error. The server ignores both tagged script that the application or session events do not use, as well as any HTML in the file.

The scripts contained in the Global.asa file may be written in any supported scripting language. If multiple event or object scripts use the same scripting language, they can be combined inside a single set of <SCRIPT> tags.

When you save changes to the Global.asa file, the server finishes processing all of the current application requests before it recompiles the Global.asa file. During that time, the server refuses additional requests and returns an error message stating that the request cannot be processed while the application is being restarted.

After all of the current user requests have been processed, the server deletes all active sessions, calling the **Session_OnEnd** event for each session it deletes, closes the application, and calls the **Application_OnEnd** event. The Global.asa file is then recompiled. Subsequent user requests will start the application and create new sessions, and trigger the **Application_OnStart** and **Session_OnStart** events.

However, saving changes to a file that is included by the Global.asa file does not cause the server to recompile Global.asa. In order for the server to recognize changes in the included file, you must once again save the Global.asa file.

Procedures declared in the Global.asa file can only be called from one or more of the scripts associated with the **Application_OnStart**, **Application_OnEnd**, **Session_OnStart**, and **Session_OnEnd** events. They are not available to the ASP pages in the ASP-based application.

To share procedures across an application, you can declare the procedures in a separate file, and then use server-side include (SSI) statements to include the file in the ASP pages that call the procedures. You typically give include files an .inc extension.

Request Object

The **Request** object retrieves the values that the client browser passed to the server during an HTTP request.

Syntax

```
Request[.collection|property|method](variable)
```

Collections

ClientCertificate	The values of fields stored in the client certificate that is sent in the HTTP request.
Cookies	The values of cookies sent in the HTTP request.
Form	The values of form elements in the HTTP request body.
QueryString	The values of variables in the HTTP query string.
ServerVariables	The values of predetermined environment variables.

Properties

TotalBytes	Read-only. Specifies the total number of bytes the client is sending in the body of the request.

Methods

BinaryRead	Retrieves data sent to the server from the client as part of a POST request.

Variable parameters are strings that specify the item to be retrieved from a collection or to be used as input for a method or property. For more information about the *variable* parameter, see the individual collection descriptions.

Remarks

If the specified variable is not in one of the preceding five collections, the **Request** object returns EMPTY.

All variables can be accessed directly by calling **Request(***variable***)** without the collection name. In this case, the Web server searches the collections in the following order.

1. **QueryString**
2. **Form**
3. **Cookies**
4. **ClientCertificate**
5. **ServerVariables**

If a variable with the same name exists in more than one collection, the **Request** object returns the first instance that the object encounters.

It is strongly recommended that when referring to members of the **ServerVariables** collection the full name be used. For example, rather than **Request**.(AUTH_USER) use **Request.ServerVariables**(AUTH_USER).

Cookies

The **Cookies** collection enables you to retrieve the values of the cookies sent in an HTTP request.

Syntax

```
Request.Cookies(cookie)[(key)|.attribute]
```

Parameters

cookie

Specifies the cookie whose value should be retrieved.

key

An optional parameter used to retrieve subkey values from cookie dictionaries.

attribute

Specifies information about the cookie itself. The attribute parameter can be the following.

Name	**Description**
HasKeys	Read-only. Specifies whether the cookie contains keys.

Remarks

You can access the subkeys of a cookie dictionary by including a value for *key*. If a cookie dictionary is accessed without specifying *key*, all of the keys are returned as a single query string. For example, if MyCookie has two keys, First and Second, and you do not specify either of these keys in a call to **Request.Cookies**, the following string is returned.

```
First=firstkeyvalue&Second=secondkeyvalue
```

If two cookies with the same name are sent by the client browser, **Request.Cookies** returns the one with the deeper path structure. For example, if two cookies had the same name but one had a path attribute of /www/ and the other of /www/home/, the client browser would send both cookies to the /www/home/ directory, but **Request.Cookies** would only return the second cookie.

To determine whether a cookie is a cookie dictionary (whether the cookie has keys), use the following script.

```
<%= Request.Cookies("myCookie").HasKeys %>
```

If myCookie is a cookie dictionary, the preceding value evaluates to TRUE. Otherwise, it evaluates to FALSE.

You can use an iterator to cycle through all the cookies in the **Cookie** collection, or all the keys in a cookie. However, iterating through keys on a cookie that does not have keys will not produce any output. You can avoid this situation by first checking to see whether a cookie has keys by using the **.HasKeys** syntax. This is demonstrated in the following example.

```
<%
'Print out the entire cookie collection.
For Each cookie in Request.Cookies
 If Not cookie.HasKeys Then
  'Print out the cookie string
%>
  <%= cookie %> = <%= Request.Cookies(cookie)%>
<%
Else
```

```
'Print out the cookie collection
For Each key in Request.Cookies(cookie)
%>
 <%= cookie %> (<%= key %>) = <%= Request.Cookies(cookie)(key)%>
<%
 Next
End If
Next
%>
```

Examples

The following example prints the value of myCookie in a Web page.

```
Here is the value of the cookie named myCookie:
<%= Request.Cookies("myCookie") %>
```

Form

The **Form** collection retrieves the values of form elements posted to the HTTP request body by a form using the POST method.

Syntax

Request.Form(*element***)[(***index***)|.Count]**

Parameters

element

Specifies the name of the form element from which the collection is to retrieve values.

index

An optional parameter that enables you to access one of multiple values for a parameter. It can be any integer in the range 1 to **Request.Form(***parameter***).Count**.

QueryString

The **QueryString** collection retrieves the values of the variables in the HTTP query string. The HTTP query string is specified by the values following the question mark (?). Several different processes can generate a query string. For example, the anchor tag

<A HREF= "example?string=this is a sample">string sample</A>

generates a variable named string with the value "this is a sample". Query strings are also generated by sending a form, or by a user typing a query into the address box of their browser.

Syntax

Request.QueryString(*variable***)[(***index***)|.Count]**

Parameters

variable

Specifies the name of the variable in the HTTP query string to retrieve.

index

An optional parameter that enables you to retrieve one of multiple values for *variable*. It can be any integer value in the range 1 to **Request.QueryString(***variable***).Count**.

Remarks

The **QueryString** collection is a parsed version of the QUERY_STRING variable in the **ServerVariables** collection. It enables you to retrieve the QUERY_STRING variables by name. The value of **Request.QueryString(***parameter***)** is an array of all of the values of *parameter* that occur in QUERY_STRING. You can determine the number of values of a parameter by calling **Request.QueryString(*parameter*).Count**. If a variable does not have multiple data sets associated with it, the count is 1. If the variable is not found, the count is 0.

To reference a **QueryString** variable in one of multiple data sets, you specify a value for *index*. The *index* parameter may be any value between 1 and **Request.QueryString(*variable*).Count**. If you reference one of multiple **QueryString** variables without specifying a value for *index*, the data is returned as a comma-delimited string.

When you use parameters with **Request.QueryString**, the server parses the parameters sent to the request and returns the specified data. If your application requires unparsed **QueryString** data, you can retrieve it by calling **Request.QueryString** without any parameters.

You can use an iterator to loop through all the data values in a query string. For example, if the following request is sent

```
http://NAMES.ASP?Q=Mert&Q=Kiran
```

and Names.asp contained the following script,

```
---NAMES.ASP---
<%
For Each item In Request.QueryString("Q")
  Response.Write item & "<BR>"
Next
%>
```

Names.asp would display the following.

```
Mert

Kiran
```

The preceding script could also have been written using **Count**.

```
<%

For I = 1 To Request.QueryString("Q").Count

 Response.Write Request.QueryString("Q")(I) & "<BR>"

Next

%>
```

Example

The client request

```
/scripts/directory-lookup.asp?name=mert&age=24
```

results in the following QUERY_STRING value.

```
name=mert&age=24.
```

The **QueryString** collection would then contain two members, name and age. You can then use the following script.

```
Welcome, <%= Request.QueryString("name") %>.

Your age is <%= Request.QueryString("age") %>.
```

The output would be

```
Welcome, Mert. Your age is 24.
```

If the following script is used

```
The unparsed query string is: <%=Request.QueryString %>
```

The output would be

```
The unparsed query string is: name=merit&age=24
```

ServerVariables

The **ServerVariables** collection retrieves the values of predetermined environment variables.

Syntax

```
Request.ServerVariables (server environment variable)
```

Parameters

server environment variable

Specifies the name of the server environment variable to retrieve. It can be one of the following values.

Variable	**Description**
ALL_HTTP	All HTTP headers sent by the client.
ALL_RAW	Retrieves all headers in the raw-form. The difference between ALL_RAW and ALL_HTTP is that ALL_HTTP places an HTTP_ prefix before the header name and the header-name is always capitalized. In ALL_RAW the header name and values appear as they are sent by the client.
APPL_MD_PATH	Retrieves the metabase path for the (WAM) Application for the ISAPI DLL.
APPL_PHYSICAL_PATH	Retrieves the physical path corresponding to the metabase path. IIS converts the APPL_MD_PATH to the physical (directory) path to return this value.

AUTH_PASSWORD	The value entered in the client's authentication dialog. This variable is only available if Basic authentication is used.
AUTH_TYPE	The authentication method that the server uses to validate users when they attempt to access a protected script.
AUTH_USER	Raw authenticated user name.
CERT_COOKIE	Unique ID for client certificate, Returned as a string. Can be used as a signature for the whole client certificate.
CERT_FLAGS	bit0 is set to 1 if the client certificate is present.
	bit1 is set to 1 if the Certifying Authority of the client certificate is invalid (not in the list of recognized CA on the server).
CERT_ISSUER	Issuer field of the client certificate (O=MS, OU=IAS, CN=user name, C=USA).
CERT_KEYSIZE	Number of bits in Secure Sockets Layer connection key size. For example, 128.
CERT_SECRETKEYSIZE	Number of bits in server certificate private key. For example, e.g. 1024.
CERT_SERIALNUMBER	Serial number field of the client certificate.
CERT_SERVER_ISSUER	Issuer field of the server certificate.
CERT_SERVER_SUBJECT	Subject field of the server certificate.
CERT_SUBJECT	Subject field of the client certificate.
CONTENT_LENGTH	The length of the content as given by the

	client.
CONTENT_TYPE	The data type of the content. Used with queries that have attached information, such as the HTTP queries GET, POST, and PUT.
GATEWAY_INTERFACE	The revision of the CGI specification used by the server. The format is CGI/revision.
HTTP_*<HeaderName>*	The value stored in the header *HeaderName*. Any header other than those listed in this table must be prefixed by HTTP_ in order for the **ServerVariables** collection to retrieve its value. **Note** The server interprets any underscore (_) characters in *HeaderName* as dashes in the actual header. For example if you specify HTTP_MY_HEADER, the server searches for a header sent as MY-HEADER.
HTTPS	Returns ON if the request came in through secure channel (SSL) or it returns OFF if the request is for a non-secure channel.
HTTPS_KEYSIZE	Number of bits in Secure Sockets Layer connection key size. For example, 128.
HTTPS_SECRETKEYSIZE	Number of bits in server certificate private key. For example, 1024.
HTTPS_SERVER_ISSUER	Issuer field of the server certificate.
HTTPS_SERVER_SUBJECT	Subject field of the server certificate.
INSTANCE_ID	The ID for the IIS instance in textual

	format. If the instance ID is 1, it appears as a string. You can use this variable to retrieve the ID of the Web-server instance (in the metabase) to which the request belongs.
INSTANCE_META_PATH	The metabase path for the instance of IIS that responds to the request.
LOCAL_ADDR	Returns the Server Address on which the request came in. This is important on multi-homed machines where there can be multiple IP addresses bound to a machine and you want to find out which address the request used.
LOGON_USER	The Windows NT® account that the user is logged into.
PATH_INFO	Extra path information as given by the client. You can access scripts by using their virtual path and the PATH_INFO server variable. If this information comes from a URL, it is decoded by the server before it is passed to the CGI script.
PATH_TRANSLATED	A translated version of PATH_INFO that takes the path and performs any necessary virtual-to-physical mapping.
QUERY_STRING	Query information stored in the string following the question mark (?) in the HTTP request.
REMOTE_ADDR	The IP address of the remote host making the request.

REMOTE_HOST	The name of the host making the request. If the server does not have this information, it will set REMOTE_ADDR and leave this empty.
REMOTE_USER	Unmapped user-name string sent in by the User. This is the name that is really sent by the user as opposed to the ones that are modified by any authentication filter installed on the server.
REQUEST_METHOD	The method used to make the request. For HTTP, this is GET, HEAD, POST, and so on.
SCRIPT_NAME	A virtual path to the script being executed. This is used for self-referencing URLs.
SERVER_NAME	The server's host name, DNS alias, or IP address as it would appear in self-referencing URLs.
SERVER_PORT	The port number to which the request was sent.
SERVER_PORT_SECURE	A string that contains either 0 or 1. If the request is being handled on the secure port, then this will be 1. Otherwise, it will be 0.
SERVER_PROTOCOL	The name and revision of the request information protocol. The format is *protocol/revision*.
SERVER_SOFTWARE	The name and version of the server software that answers the request and runs the gateway. The format is *name/version*.

URL	Gives the base portion of the URL.

Remarks

If a client sends a header other than those specified in the preceding table, you can retrieve the value of that header by prefixing the header name with HTTP_ in the call to **Request.ServerVariables**. For example, if the client sent this header

```
SomeNewHeader:SomeNewValue
```

you could retrieve SomeNewValue by using the following syntax

```
<% Request.ServerVariables("HTTP_SomeNewHeader") %>
```

You can use an iterator to loop through each server variable name. For example, the following script prints out all of the server variables in a table.

```
<TABLE>

<TR><TD><B>Server Variable</B></TD><TD><B>Value</B></TD></TR>

<% For Each name In Request.ServerVariables %>

<TR><TD> <%= name %> </TD><TD>  <%= Request.ServerVariables(name) %> </TD></TR>

</TABLE>

<% Next %>
```

Example

The following example uses the **Request** object to display several server variables.

```
<HTML>

<!-- This example displays the content of several ServerVariables. -->

ALL_HTTP server variable =
<%= Request.ServerVariables("ALL_HTTP") %> <BR>

CONTENT_LENGTH server variable =

<%= Request.ServerVariables("CONTENT_LENGTH") %> <BR>

CONTENT_TYPE server variable =

<%= Request.ServerVariables("CONTENT_TYPE") %> <BR>

QUERY_STRING server variable =

<%= Request.ServerVariables("QUERY_STRING") %> <BR>

SERVER_SOFTWARE server variable =

<%= Request.ServerVariables("SERVER_SOFTWARE") %> <BR>

</HTML>
```

The next example uses the **ServerVariables** collection to insert the name of the server into a hyperlink.

```
<A HREF = "http://<%= Request.ServerVariables("SERVER_NAME") %>

/scripts/MyPage.asp">Link to MyPage.asp</A>
```

Response Object

You can use the **Response** object to send output to the client.

Syntax

Response.*collection|property|method*

Collections

Cookies	Specifies cookie values. Using this collection, you can set cookie values.

Properties

Buffer	Indicates whether page output is buffered.
CacheControl	Determines whether proxy servers are able to cache the output generated by ASP.
Charset	Appends the name of the character set to the content-type header.
ContentType	Specifies the HTTP content type for the response.
Expires	Specifies the length of time before a page cached on a browser expires.
ExpiresAbsolute	Specifies the date and time on which a page cached on a browser expires.
IsClientConnected	Indicates whether the client has disconnected from the server.
Pics	Adds the value of a PICS label to the pics-label field of the response header.
Status	The value of the status line returned by the server.

Methods

AddHeader	Sets the HTML header *name* to *value*.
AppendToLog	Adds a string to the end of the Web server log entry for this request.
BinaryWrite	Writes the given information to the current HTTP output without any character-set conversion.
Clear	Erases any buffered HTML output.
End	Stops processing the .asp file and returns the current result.
Flush	Sends buffered output immediately.

Redirect	Sends a redirect message to the browser, causing it to attempt to connect to a different URL.
Write	Writes a variable to the current HTTP output as a string.

Cookies

The **Cookies** collection sets the value of a cookie. If the specified cookie does not exist, it is created. If the cookie exists, it takes the new value and the old value is discarded.

Syntax

Response.Cookies(*cookie***)[(***key***)|.***attribute***] =** *value*

Parameters

cookie

The name of the cookie.

key

An optional parameter. If *key* is specified, cookie is a dictionary, and *key* is set to *value*.

attribute

Specifies information about the cookie itself. The attribute parameter can be one of the following.

Name	**Description**
Domain	Write-only. If specified, the cookie is sent only to requests to this domain.
Expires	Write-only. The date on which the cookie expires. This date must be set in order for the cookie to be stored on the client's disk after the session ends. If this attribute is not set to a date beyond the current date, the cookie will expire when the session ends.

HasKeys	Read-only. Specifies whether the cookie contains keys.
Path	Write-only. If specified, the cookie is sent only to requests to this path. If this attribute is not set, the application path is used.
Secure	Write-only. Specifies whether the cookie is secure.

Value

Specifies the value to assign to *key* or *attribute*.

Remarks

If a cookie with a key is created, as in the following script,

```
<%
Response.Cookies("mycookie")("type1") = "NEU"
Response.Cookies("mycookie")("type2") = "COMPUTER ENGINEERING"
%>
```

this header is sent.

```
Set-Cookie:MYCOOKIE=TYPE1=NEU&TYPE2=COMPUTER+ENGINEERING
```

A subsequent assignment to myCookie without specifying a key, would destroy type1 and type2. This is shown in the following example.

```
<% Response.Cookies("myCookie") = "chocolate chip" %>
```

In the preceding example, the keys type1 and type2 are destroyed and their values are discarded. The myCookie cookie now has the value chocolate chip.

Conversely, if you call a cookie with a key, it destroys any nonkey values the cookie contained. For example, if after the preceding code you call **Response.Cookies** with the following

```
<% Response.Cookies("myCookie")("newType") = "peanut butter" %>
```

The value chocolate chip is discarded and newType would be set to peanut butter.

To determine whether a cookie has keys, use the following syntax.

```
<%= Response.Cookies("myCookie").HasKeys %>
```

If myCookie is a cookie dictionary, the preceding value is TRUE. Otherwise, it is FALSE.

You can use an iterator to set cookie attributes. For example, to set all of the cookies to expire on a particular date, use the following syntax.

```
<%
For Each cookie in Response.Cookies
 Response.Cookie(cookie).Expires = #July 4, 1997#
Next
%>
```

You can also use an iterator to set the values of all the cookies in a collection, or all the keys in a cookie. However, the iterator, when invoked on a cookie that does not have keys, does not execute. To avoid this, you can first use the **.HasKeys** syntax to check whether a cookie has any keys. This is demonstrated in the following example.

```
<%
If Not cookie.HasKeys Then
 'Set the value of the cookie
```

```
Response.Cookies(cookie) = ""
Else
 'Set the value for each key in the cookie collection
 For Each key in Response.Cookies(cookie)
   Response.Cookies(cookie)(key) = ""
 Next key
%>
```

Examples

The following examples demonstrate how you can set a value for a cookie and assign values to its attributes.

```
<%
Response.Cookies("Type") = "Chocolate Chip"
Response.Cookies("Type").Expires = "July 31, 1997"
Response.Cookies("Type").Domain = "msn.com"
Response.Cookies("Type").Path = "/www/home/"
Response.Cookies("Type").Secure = FALSE%>
```

Charset

The **Charset** property appends the name of the character set (for example, ISO-LATIN-7) to the content-type header in the response object.

Syntax

Response.Charset(*CharsetName*)

Parameters

CharsetName

A string that specifies a character set for the page. The character set name will be appended to the content-type header in the **Response** object.

Example

For an ASP page that did not include the **Response.Charset** property, the content-type header would be.

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

If the same .asp file included

```
<% Response.Charset("ISO-LATIN-7") %>
```

the content-type header would be

```
content-type:text/html; charset=ISO-LATIN-7
```

Remarks

This function inserts any string in the header, whether it represents a valid character set or not.

If a single page contains multiple tags containing **Response.Charset**, each **Response.Charset** will replace the previous *CharsetName*. As a result, the character set will be set to the value specified by the last instance of **Response.Charset** in the page.

On the Macintosh, the default U.S. character set is not ISO-LATIN-1. When serving up documents, Personal Web Server for Macintosh automatically converts from the Macintosh character set to ISO-Latin-1. In the U.S. version, all pages are assumed to be in the U.S. Macintosh character set unless the **Response.Charset** is used. If **Response.Charset** is used to change the character set, Personal Web Server for Macintosh does not convert the character set.

Expires

The **Expires** property specifies the length of time before a page cached on a browser expires. If the user returns to the same page before it expires, the cached version is displayed.

Syntax

```
Response.Expires [= number]
```

Parameters

number

The time in minutes before the page expires. Set this parameter to 0 to have the cached page expire immediately.

Remarks

If this property is set more than once on a page, the shortest time is used.

ExpiresAbsolute

The **ExpiresAbsolute** property specifies the date and time at which a page cached on a browser expires. If the user returns to the same page before that date and time, the cached version is displayed. If a time is not specified, the page expires at midnight of that day. If a date is not specified, the page expires at the given time on the day that the script is run.

Syntax

```
Response.ExpiresAbsolute [= [date] [time]]
```

Parameters

date

Specifies the date on which the page will expire. The value sent in the expires header conforms to the RFC-1123 date format.

time

Specifies the time at which the page will expire. This value is converted to GMT before an Expires header is sent.

Remarks

If this property is set more than once on a page, the earliest expiration date or time is used.

Example

The following example specifies that the page will expire 15 seconds after 1:30 pm on May 31, 2003.

```
<% Response.ExpiresAbsolute=#May 31,2003 13:30:15# %>
```

Status

The **Status** property specifies the value of the status line returned by the server. Status values are defined in the HTTP specification.

Syntax

```
Response.Status = StatusDescription
```

Parameters

StatusDescription

A string that consists of both a three-digit number that indicates a status code and a brief explanation of that code. For example, 310 Move Permanently.

Remarks

Use this property to modify the status line returned by the server.

The following example sets the response status.

```
<% Response.Status = "401 Unauthorized" %>
```

CHAPTER 3: ASP Methods

AddHeader

The **AddHeader** method adds an HTML header with a specified value. This method always adds a new HTTP header to the response. It will not replace an existing header of the same name. Once a header has been added, it cannot be removed.

This method is for advanced use only. If another **Response** method will provide the functionality you require, it is recommended that you use that method instead.

Syntax

```
Response.AddHeader name, value
```

Parameters

name

The name of the new header variable.

value

The initial value stored in the new header variable.

Remarks

To avoid name ambiguity, *name* should not contain any underscore (_) characters. The **ServerVariables** collection interprets underscores as dashes in the header name. For example, the following script causes the server to search for a header named MY-HEADER.

```
<% Request.ServerVariables("HTTP_MY_HEADER") %>
```

Because HTTP protocol requires that all headers be sent before content, you must call **AddHeader** in your script before any output (such as that generated by HTML code or the **Write** method) is sent to the client. The exception to this rule is when the **Buffer** property is set to TRUE. If the output is buffered, you can call the **AddHeader** method at any point in the script, as long as it precedes any calls to **Flush**. Otherwise, the call to **AddHeader** will generate a run-time error.

Clear

The **Clear** method erases any buffered HTML output. However, the **Clear** method only erases the response body; it does not erase response headers. You can use this method to handle error cases. Note that this method will cause a run-time error if **Response.Buffer** has not been set to TRUE.

Syntax

```
Response.Clear
```

End

The **End** method causes the Web server to stop processing the script and return the current result. The remaining contents of the file are not processed.

Syntax

```
Response.End
```

Remarks

If **Response.Buffer** has been set to TRUE, calling **Response.End** will flush the buffer. If you do not want output returned to the user, you should call

```
<%
Response.Clear
Response.End
%>
```

Flush

The **Flush** method sends buffered output immediately. This method will cause a run-time error if **Response.Buffer** has not been set to TRUE.

Syntax

```
Response.Flush
```

Remarks

If the **Flush** method is called on an ASP page, the server does not honor Keep-Alive requests for that page.

Redirect

The **Redirect** method causes the browser to attempt to connect to a different URL.

Syntax

```
Response.Redirect URL
```

Parameters

URL

The Uniform Resource Locator that the browser is redirected to.

Remarks

Any response body content set explicitly in the page is ignored. However, this method does send other HTTP headers set by this page to the client. An automatic response body containing the redirect URL as a link is generated. The **Redirect** method sends the following explicit header, where *URL* is the value passed to the method.

```
HTTP/1.0 302 Object Moved

Location URL
```

Write

The **Write** method writes a specified string to the current HTTP output.

Syntax

```
Response.Write variant
```

Parameters

variant

The data to write. This parameter can be any data type supported by the Visual Basic® Scripting Edition VARIANT data type, including characters, strings, and integers. This value cannot contain the character combination %>; instead you should use the escape sequence %\>. The Web server will translate the escape sequence when it processes the script.

Examples

The following examples use the **Response.Write** method to send output to the client.

```
My Graduation Project topic is <% Response.Write "Medical Consultancy with ASP." %>

Your name is: <% Response.Write Request.Form("name") %>
```

The following example adds an HTML tag to the Web page output. Because the string returned by the **Write** method cannot contain the character combination %>, the escape %\> has been used instead. The following script

```
<% Response.Write "<TABLE WIDTH = 100%\>" %>
```

produces the output

```
<TABLE WIDTH = 100%>
```

Server Object

The **Server** object provides access to methods and properties on the server. Most of these methods and properties serve as utility functions.

Syntax

Server.*property|method*

Properties

ScriptTimeout	The amount of time that a script can run before it times out.

Methods

CreateObject	Creates an instance of a server component.
HTMLEncode	Applies HTML encoding to the specified string.
MapPath	Maps the specified virtual path, either the absolute path on the current server or the path relative to the current page, into a physical path.
URLEncode	Applies URL encoding rules, including escape characters, to the string.

ScriptTimeout

The **ScriptTimeout** property specifies the maximum amount of time a script can run before it is terminated.

The timeout will not take effect while a server component is processing.

Syntax

```
Server.ScriptTimeout = NumSeconds
```

Parameters

NumSeconds

Specifies the maximum number of seconds that a script can run before the server terminates it. The default value is 90 seconds.

Remarks

A default **ScriptTimeout** can be set for a Web Service or Web Server by using the **AspScriptTimeout** property in the metabase. The **ScriptTimeout** property cannot be set to a value less than that specified in the metabase. For example, if *NumSeconds* is set to 10, and the metabase setting contains the default value of 90 seconds, scripts will time out after 90 seconds. However, if *NumSeconds* were set to 100, the scripts would time out after 100 seconds.

For more information about using the metabase, see **About the Metabase**

Example

The following example causes scripts to time out if the server takes longer than 100 seconds to process them.

```
<% Server.ScriptTimeout = 100 %>
```

The following example retrieves the current value of the **ScriptTimeout** property and stores it in the variable TimeOut.

```
<% TimeOut = Server.ScriptTimeout %>
```

CreateObject

The **CreateObject** method creates an instance of a server component. If the component has implemented the **OnStartPage** and **OnEndPage** methods, the **OnStartPage** method is called at this time. For more information about server components, see **Installable Components for ASP**.

Syntax

```
Server.CreateObject( progID )
```

Parameters

progID

Specifies the type of object to create. The format for *progID* is [*Vendor.*]*Component*[*.Version*].

Remarks

By default, objects created by the **Server.CreateObject** method have page scope. This means that they are automatically destroyed by the server when it finishes processing the current ASP page.

To create an object with session or application scope, you can either use the <OBJECT> tag and set the SCOPE attribute to SESSION or APPLICATION, or store the object in a session or application variable.

For example, an object stored in a session variable, as shown in the following script, is destroyed when the **Session** object is destroyed. That is, when the session times out, or the **Abandon** method is called.

```
<% Set Session("ad") = Server.CreateObject("MSWC.AdRotator")%>
```

You can also destroy the object by setting the variable to Nothing or setting the variable to a new value, as shown below. The first example releases the object ad. The second replaces ad with a string.

```
<% Session("ad") = Nothing %>

<% Session("ad") = "some other value" %>
```

You cannot create an object instance with the same name as a built-in object. For example, the following returns an error.

```
<% Set Response = Server.CreateObject("Response") %>
```

Example

```
<% Set MyAd = Server.CreateObject("MSWC.AdRotator") %>
```

The preceding example creates a server component, MyAd, as a **MSWC.AdRotator** component that can be used to automate rotation of advertisements on a Web page.

Session Object

You can use the **Session** object to store information needed for a particular user-session. Variables stored in the **Session** object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire user-session.

The Web server automatically creates a **Session** object when a Web page from the application is requested by a user who does not already have a session. The server destroys the **Session** object when the session expires or is abandoned.

One common use for the **Session** object is to store user preferences. For example, if a user indicates that they prefer not to view graphics, you could store that information in the **Session** object. For more information on using the **Session** object, see Managing Sessions in the ASP Applications section.

Note Session state is only maintained for browsers that support cookies.

Syntax

```
Session.collection|property|method
```

Collections

Contents	Contains the items that you have added to the session with script commands.
StaticObjects	Contains the objects created with the <OBJECT> tag and given session scope.

Properties

CodePage	The codepage that will be used for symbol mapping.
LCID	The locale identifier.
SessionID	Returns the session identification for this user.
Timeout	The timeout period for the session state for this application, in minutes.

Methods

Abandon	This method destroys a **Session** object and releases its resources.

Events

Scripts for the following events are declared in the global.asa file.

Session_OnEnd

Session_OnStart

For more information about these events and the global.asa file, see the **Global.asa Reference**.

SessionID

The **SessionID** property returns the session identification for this user. Each session has a unique identifier that is generated by the server when the session is created. The session ID is returned as a **LONG** data type.

Syntax

```
Session.SessionID
```

Remarks

You should not use the **SessionID** property to generate primary key values for a database application. This is because if the Web server is restarted, some **SessionID** values may be the same as those generated before the server was stopped. Instead, you should use an autoincrement column data type, such as IDENTITY with Microsoft® SQL Server, or COUNTER with Microsoft® Access.

Timeout

The **Timeout** property specifies the timeout period assigned to the **Session** object for this application, in minutes. If the user does not refresh or request a page within the timeout period, the session ends.

Syntax

```
Session.Timeout [ = nMinutes]
```

Parameters

nMinutes

Specifies the number of minutes that a session can remain idle before the server terminates it automatically. The default is 20 minutes.

CONCLUSION

Visual Basic is known for its ability to allow a programmer to quickly, and easily, create a Windows program. In part, this is possible because objects provide code that you don't have to write.

In Visual Basic, the basic building block of an application is a form, which is simply a window. The Visual Basic IDE can insert forms into your project, and then you can resize the forms as well as change other properties of the form. However, controls (checkboxes, textboxes...) are also windows. A form is distinguished from a control in that only forms can exist as standalone objects. When controls are used, they must be placed in a form and there are a few exceptions such as the printer object or the screen object, which are not considered part of any form, but are part of a Visual Basic program.

It's worth noting that when managers talk about programmers, one of the common metrics used to describe performance is "lines of code per month". There are all kinds of debate about how good a metric this is, but the fact is that the metric is used.

It's not that you don't get credit for novel algorithms, or that you won't be a hero to fellow programmers when they see how you solved a problem with 10 lines of code that took them 100. You'll get that credit (and mental satisfaction) but looking at the big picture it's clear that the volume of code you can crank out will be the visible result of your efforts. Writing a program is essentially a design task, aimed at solving a problem. Like most things, there are many design approaches for every problem. Some solve the problem by grinding out an answer. Some are very elegant. Others are so complex that even the designer has a hard time keeping up with the convolutions of the approach. Here the specifications of a great Visual Basic programmer:

1. Great programmers must be coding experts
2. Great programmers understand that there are many approaches to a problem
3. Great programmers work efficiently.

VB provides two controls, which make the link to the database file, and which creates the recordset that is exposed to the rest of the controls in your application. The two are identical in concept but differ in the flexibility they offer to the programmer. Data controls are the most important part of the programs, and applications. By adding user's information on a database file, the programmers can use this file by adding, reading and deleting fields on it.

In case you have any question about the value of objects, that objects (in the form of controls) are the single biggest reason why Visual Basic has been so successful. In Visual Basic 6.0, Microsoft has expanded the object features of Visual Basic, making it even more powerful than ever.

This Internet café reservation program contains user-friendly forms, some API's, Activex controls on it to fortify the application. It provides sensible security and usability for users and operators. The structure of program is built up on to client-server relation. Internet café's finds appropriate solution to their security or automation problems, but human relation between users and operators still important in the enterprises.

This project uses some Internet café reservation and automation techniques. It supports security and automation advantage to manage calculation of charges and reservation transactions. This program provides these possibilities with use Visual Basic functions, database controls and other many resources of Visual Basic.

REFERENCES

[1] Lecture Notes of Com 340 given by Mr Ümit İLHAN, 2001

[2] Zafer DEMİRKOL, ASP ile Web Programcılığı ve Elektronik Ticaret, Pusula Yayınları, Beyoğlu, İSTANBUL, 2002

[3] Active Server Pages:Platform SDK., Microsoft Help File, 2002

[4] www.msdn.microsoft.com/library, Microsoft's official library site, 2003

[5] ASP.NET, Web Developers' Guide, Syngress, USA, 2002

[6] Introduction to ASP, AppDev Products Company, LLC, 2001

[7] Microsoft Asp Developers Site, Retrieved 2002 from World Wide Web:http://www.Microsoft.com/asp/.

[8] www.asptoday.com, forum pages, Retrieved 2002 from web.

[9] Ferhat KARATAŞ, Web Tasarımı ve E-iş Modelleme: BilgeAdam Bilgisayar Eğitim Merkezi, BilgeAdam Yayınları., İstanbul, 2002

[10] aspnedir.com, web pages, Retrieved 2002 from world wide web

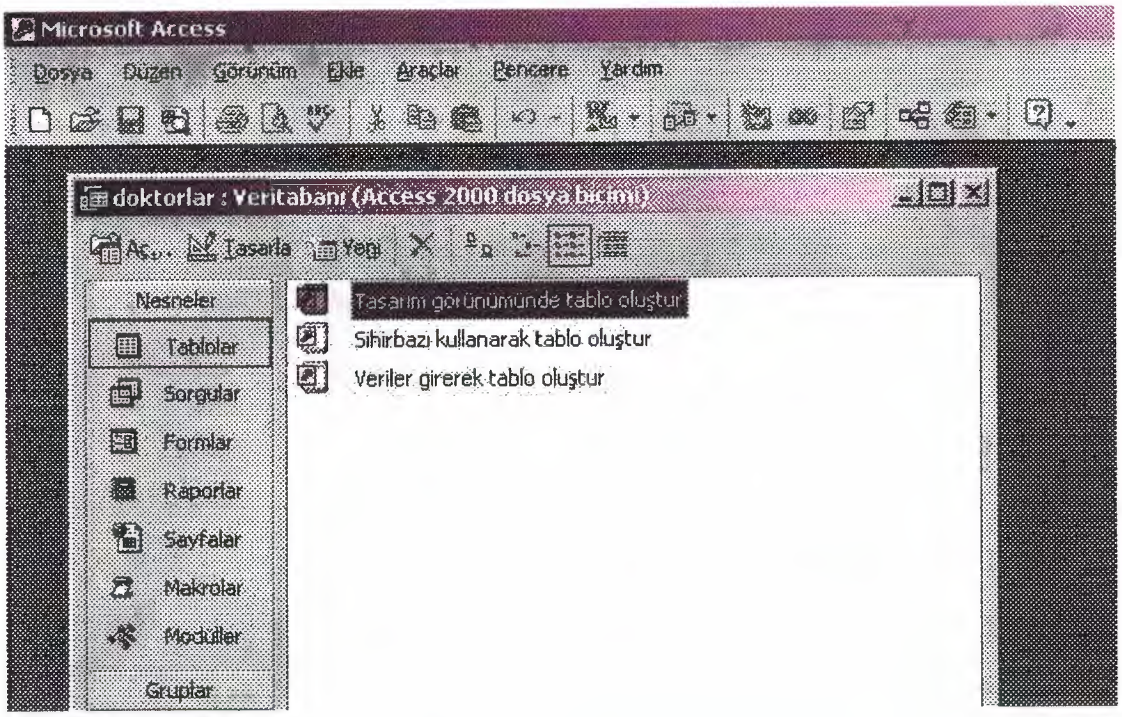
[11] aspindir.com, web pages, Retrieved 2002 from world wide web

Step by step preparation project.

1. Preparing database.
2. sharing datas
3. asp codes
4. html design
5. publishing

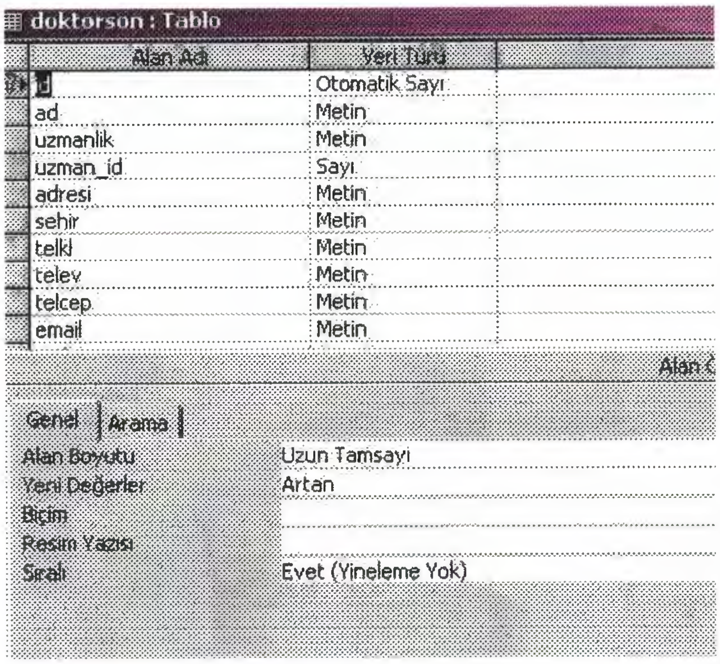
As I wrote in my introduction, preparing database is very important in my project. At first I decided which database program I will use. I chose Microsoft Access Database, because my project's main part is ASP is Microsoft's developed language. Also Access is Microsoft's database program and both of them are well adjusted each other. For preparing database first step is collecting data. After collecting datum I started to use them on Access. Here I selected for starting new database top of the list.(Fig. 1.1)

Fig.1.1


Microsoft Access
Dosya Düzen Görünüm Ekle Araçlar Pencere Yardım
doktorlar : Veritabanı (Access 2000 dosya biçimi)
Aç... Tasarla Yeni
Nesneler
Tablolar
Sorgular
Formlar
Raporlar
Sayfalar
Makrolar
Modüller
Gruplar
Tasarım görünümünde tablo oluştur
Sihirbazı kullanarak tablo oluştur
Veriler girerek tablo oluştur

Then I wrote my fields which I will enter data.(Fig.1.2.)

Fig.1.2.


doktorson : Tablo

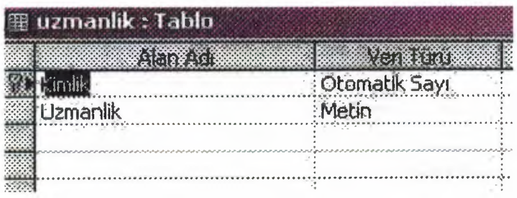
Alan Adı	Veri Türü
id	Otomatik Sayı
ad	Metin
uzmanlik	Metin
uzman_id	Sayı
adresi	Metin
sehir	Metin
telkl	Metin
telev	Metin
telcep	Metin
email	Metin

Genel | Arama

Alan Boyutu	Uzun Tamsayı
Yeni Değerler	Artan
Biçim	
Resim Yazısı	
Sıralı	Evet (Yineleme Yok)

For my project as I have decided before, I need many tables for dividing and collecting data. If I wrote everything in one table it may take more time. So I prepared new table which name is uzmanlik, and it contains expert groups for doctors. It just have two columns.(Fig.1.3.)

Fig.1.3.



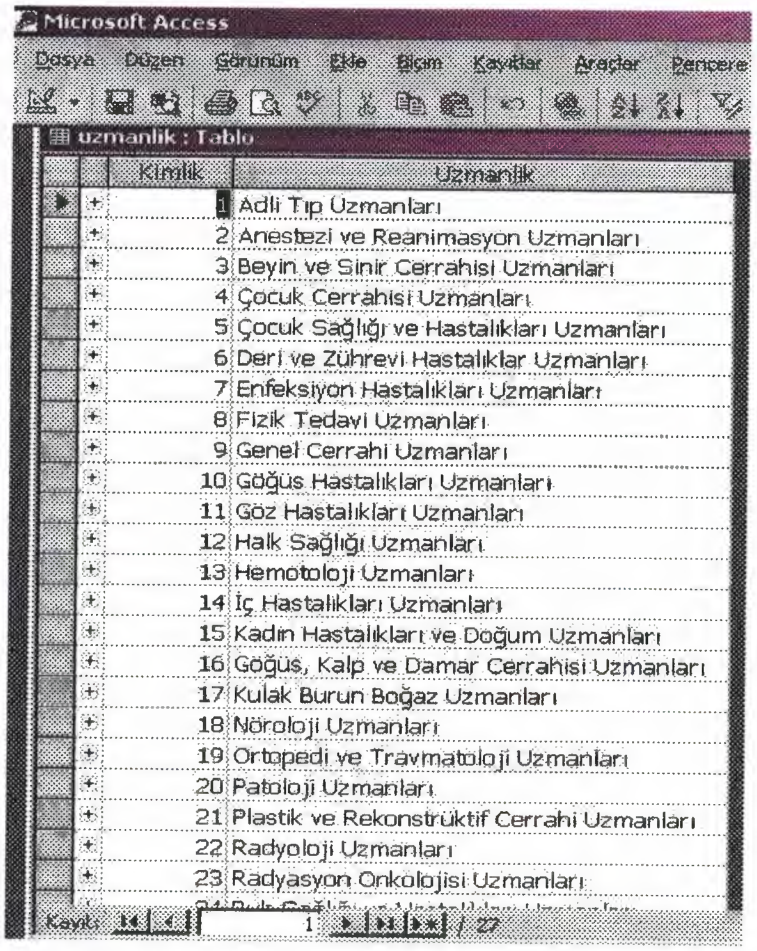
uzmanlik : Tablo

Alan Adı	Veri Türü
Kimlik	Otomatik Sayı
Uzmanlik	Metin

Here I want to explain you one important thing. You can see in below figure(1.3.) left side of kimlik there is a key. This is primary key. When we prepare table you need one primary key. It's default type is not allowed for making some changes in this and you cant use that part more than one times. It usually used for id numbers.

I prepared tables for also eczaneler(pharmacies), disciler(dentists) and hastalıklar(diseas). After this point I start to use these tables with my data. I entered all data what I need. For uzmanlik(proficiency) I grouped doctors' departs.(Fig.1.4.)

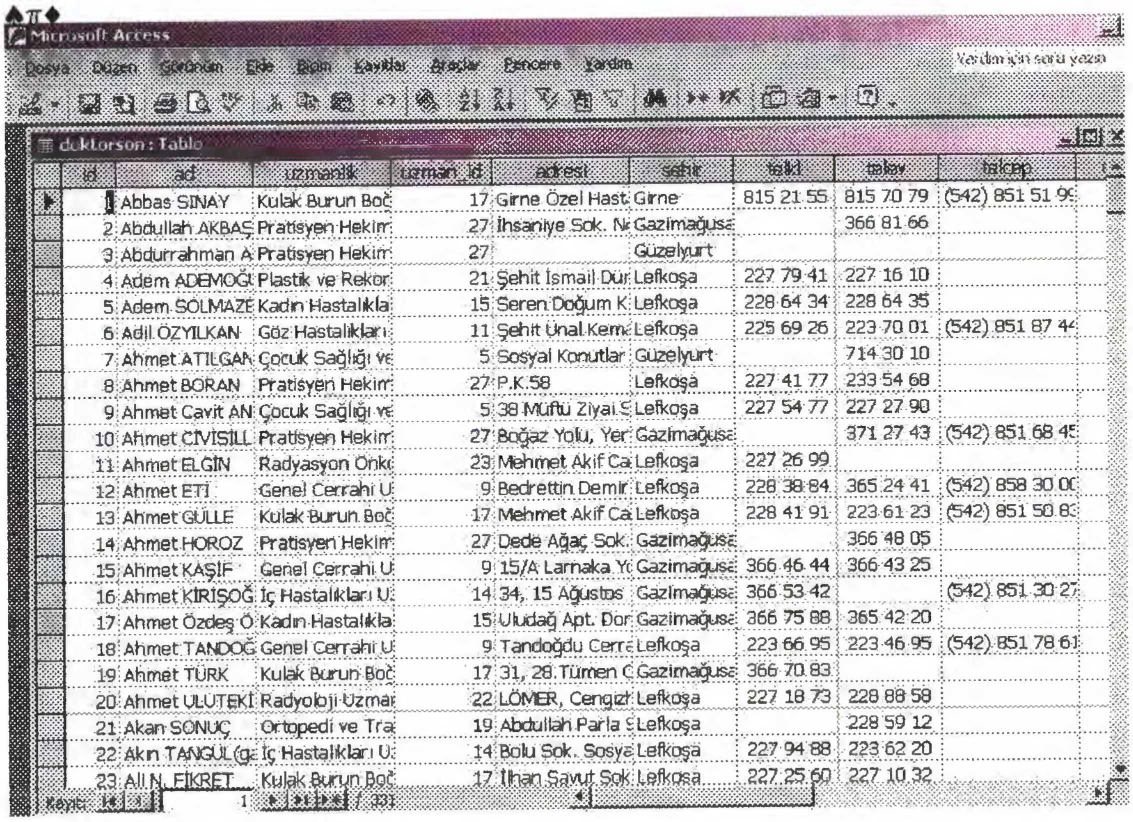
Fig.1.4.



Kimlik	Uzmanlık
1	Adli Tıp Uzmanları
2	Anestezi ve Reanimasyon Uzmanları
3	Beyin ve Sinir Cerrahisi Uzmanları
4	Çocuk Cerrahisi Uzmanları
5	Çocuk Sağlığı ve Hastalıkları Uzmanları
6	Deri ve Zührevi Hastalıklar Uzmanları
7	Enfeksiyon Hastalıkları Uzmanları
8	Fizik Tedavi Uzmanları
9	Genel Cerrahi Uzmanları
10	Göğüs Hastalıkları Uzmanları
11	Göz Hastalıkları Uzmanları
12	Halk Sağlığı Uzmanları
13	Hemotoloji Uzmanları
14	İç Hastalıkları Uzmanları
15	Kadın Hastalıkları ve Doğum Uzmanları
16	Göğüs, Kalp ve Damar Cerrahisi Uzmanları
17	Kulak Burun Boğaz Uzmanları
18	Nöroloji Uzmanları
19	Ortopedi ve Travmatoloji Uzmanları
20	Patoloji Uzmanları
21	Plastik ve Rekonstrüktif Cerrahi Uzmanları
22	Radyoloji Uzmanları
23	Radyasyon Onkolojisi Uzmanları
24	[illegible]

Then I prepared the most important part of my project, doktor's table which includes all doctors' information in Turkish Republic Of North Cyprus.(Fig.1.5.)

Fig.1.5.

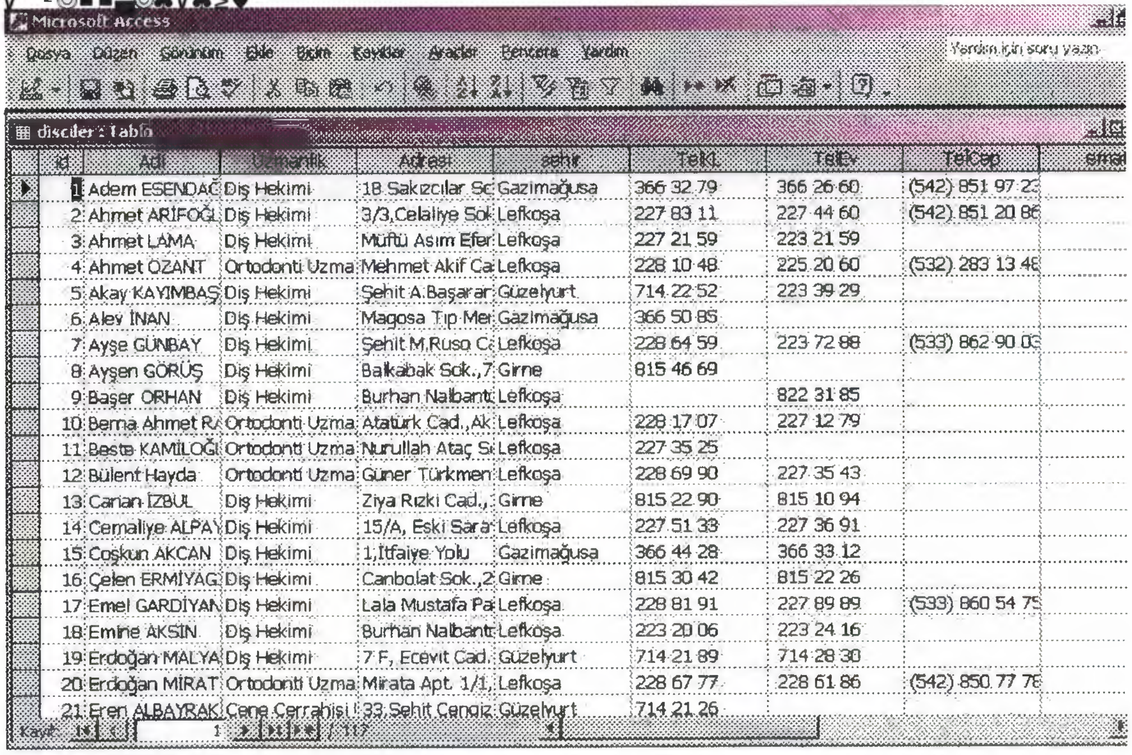


id	ad	uzmanlık	uzman_id	adresi	sehir	telid	telev	telcep
1	Abbas SINAY	Kulak Burun Boğ	17	Girne Özel Hast	Girne	815 21 55	815 70 79	(542) 851 51 9
2	Abdullah AKBAŞ	Pratisyen Hekim	27	İhsaniye Sok. N	Gazimağusa		366 81 66	
3	Abdurrahman A	Pratisyen Hekim	27		Güzelyurt			
4	Adem ADEMOĞL	Plastik ve Rekor	21	Şehit İsmail Dür	Lefkoşa	227 79 41	227 16 10	
5	Adem SOLMAZE	Kadın Hastalıkla	15	Seren Doğum K	Lefkoşa	228 64 34	228 64 35	
6	Adil ÖZYILKAN	Göz Hastalıkları	11	Şehit Ünal Kem	Lefkoşa	225 69 26	223 70 01	(542) 851 87 4
7	Ahmet ATILGAN	Çocuk Sağlığı ve	5	Sosyal Konutlar	Güzelyurt		714 30 10	
8	Ahmet BORAN	Pratisyen Hekim	27	P.K.58	Lefkoşa	227 41 77	233 54 68	
9	Ahmet Cavit AN	Çocuk Sağlığı ve	5	38 Müftü Ziyai S	Lefkoşa	227 54 77	227 27 90	
10	Ahmet CİVİSİLL	Pratisyen Hekim	27	Boğaz Yolu, Yer	Gazimağusa		371 27 43	(542) 851 68 4
11	Ahmet ELGİN	Radyasyon Onk	23	Mehmet Akif Ca	Lefkoşa	227 26 99		
12	Ahmet ETİ	Genel Cerrahi U	9	Bedrettin Demir	Lefkoşa	228 38 84	365 24 41	(542) 858 30 0
13	Ahmet GÜLLE	Kulak Burun Boğ	17	Mehmet Akif Ca	Lefkoşa	228 41 91	223 61 23	(542) 851 50 8
14	Ahmet HOROZ	Pratisyen Hekim	27	Dede Ağaç Sok.	Gazimağusa		366 48 05	
15	Ahmet KAŞİF	Genel Cerrahi U	9	15/A Larnaka Y	Gazimağusa	366 46 44	366 43 25	
16	Ahmet KİRİŞOĞ	İç Hastalıkları U	14	34, 15 Ağustos	Gazimağusa	366 53 42		(542) 851 30 2
17	Ahmet Özdeş Ö	Kadın Hastalıkla	15	Uludağ Apt. Dor	Gazimağusa	366 75 88	365 42 20	
18	Ahmet TANDOĞ	Genel Cerrahi U	9	Tandoğdu Cerr	Lefkoşa	223 66 95	223 46 95	(542) 851 78 6
19	Ahmet TÜRK	Kulak Burun Boğ	17	31, 28.Tümen C	Gazimağusa	366 70 83		
20	Ahmet ULUTEKİ	Radyoloji Uzma	22	LÖMER, Cengiz	Lefkoşa	227 18 73	228 88 58	
21	Akan SONUÇ	Ortopedi ve Tra	19	Abdullah Parla S	Lefkoşa		228 59 12	
22	Akın TANGÜL(g	İç Hastalıkları U	14	Bolu Sok. Sosya	Lefkoşa	227 94 88	223 62 20	
23	Ali N. FİKRET	Kulak Burun Boğ	17	İlhan Savut Sok	Lefkoşa	227 25 60	227 10 32	

As I prepared this, like same I prepared a table for dentists(Fig.1.6.)

Fig.1.6.

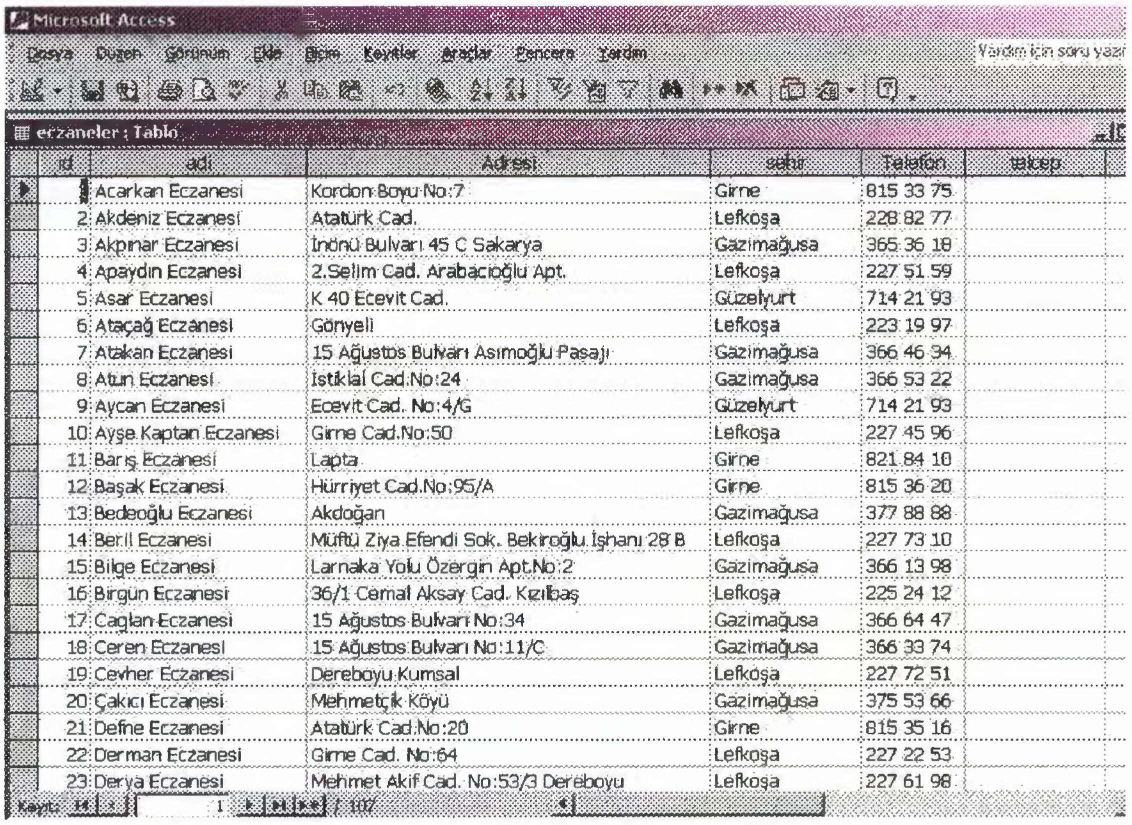
√ 2☺■■_☺♠√♠≥♦



id	Adi	Uzmanlık	Adresi	şehir	TelKL	TelEv	TelCep	emai
1	Adem ESENDAĞ	Diş Hekimi	18 Sakızcılar Sc	Gazimağusa	366 32 79	366 26 60	(542) 851 97 2	
2	Ahmet ARİFOĞL	Diş Hekimi	3/3,Celaliye Sok	Lefkoşa	227 83 11	227 44 60	(542) 851 20 8	
3	Ahmet LAMA	Diş Hekimi	Müftü Asım Efer	Lefkoşa	227 21 59	223 21 59		
4	Ahmet ÖZANT	Ortodonti Uzma	Mehmet Akif Ca	Lefkoşa	228 10 48	225 20 60	(532) 283 13 4	
5	Akay KAYIMBAŞ	Diş Hekimi	Şehit A.Başarar	Güzelyurt	714 22 52	223 39 29		
6	Alev İNAN	Diş Hekimi	Magosa Tıp Mer	Gazimağusa	366 50 85			
7	Ayşe GÜNBAY	Diş Hekimi	Şehit M.Ruso C	Lefkoşa	228 64 59	223 72 88	(533) 862 90 0	
8	Ayşen GÖRÜŞ	Diş Hekimi	Balkabak Sok.,7	Girne	815 46 69			
9	Başer ORHAN	Diş Hekimi	Burhan Nalbant	Lefkoşa		822 31 85		
10	Berna Ahmet R	Ortodonti Uzma	Atatürk Cad.,Ak	Lefkoşa	228 17 07	227 12 79		
11	Beste KAMİLOĞ	Ortodonti Uzma	Nurullah Ataç S	Lefkoşa	227 35 25			
12	Bülent Hayda	Ortodonti Uzma	Güner Türkmen	Lefkoşa	228 69 90	227 35 43		
13	Canan İZBUL	Diş Hekimi	Ziya Rızki Cad.,	Girne	815 22 90	815 10 94		
14	Cemaliye ALPAY	Diş Hekimi	15/A, Eski Sara	Lefkoşa	227 51 33	227 36 91		
15	Coşkun AKCAN	Diş Hekimi	1,İtfaiye Yolu	Gazimağusa	366 44 28	366 33 12		
16	Çelen ERMİYAG	Diş Hekimi	Canbolat Sok.,2	Girne	815 30 42	815 22 26		
17	Emel GARDİYAN	Diş Hekimi	Lala Mustafa Pa	Lefkoşa	228 81 91	227 89 89	(533) 860 54 7	
18	Emine AKSİN	Diş Hekimi	Burhan Nalbant	Lefkoşa	223 20 06	223 24 16		
19	Erdoğan MALYA	Diş Hekimi	7 F, Ecevit Cad.	Güzelyurt	714 21 89	714 28 30		
20	Erdoğan MİRAT	Ortodonti Uzma	Mirata Apt. 1/1,	Lefkoşa	228 67 77	228 61 86	(542) 850 77 7	
21	Eren ALBAYRAK	Çene Cerrahisi	33,Şehit Cengiz	Güzelyurt	714 21 26			

for preparing pharmacies, I used same technique(Fig.1.7.)

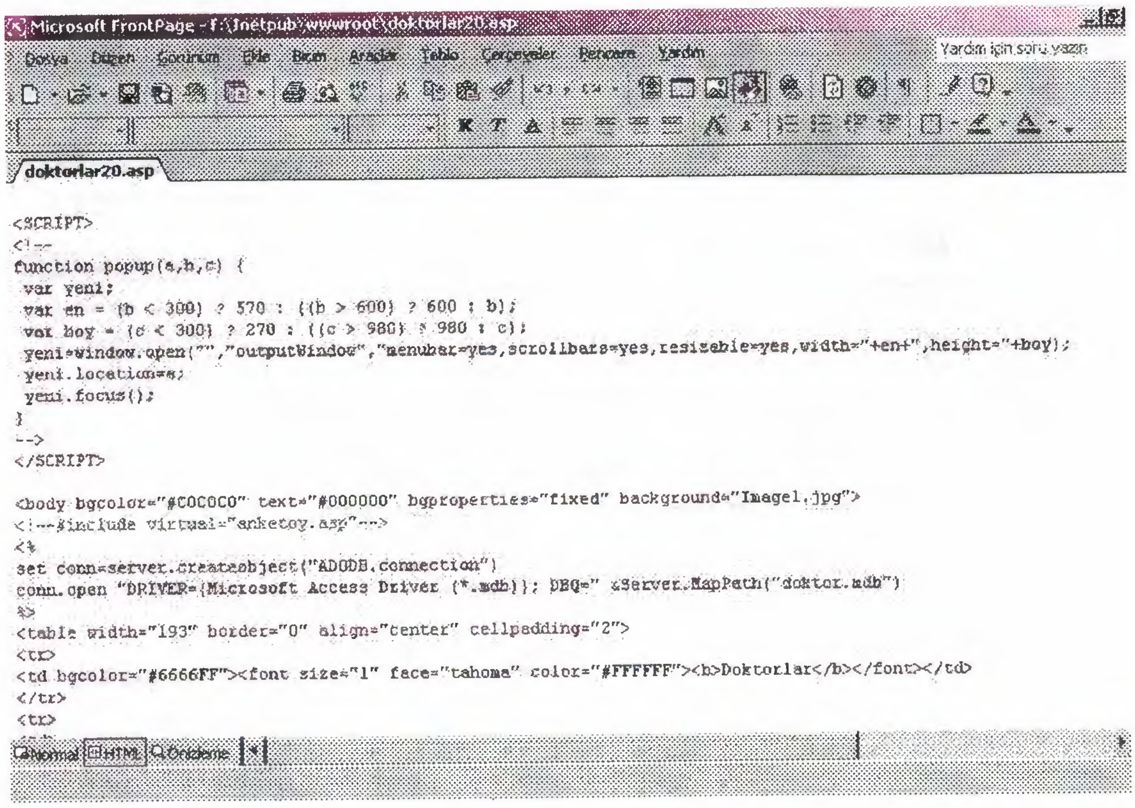
Fig.1.7.



id	adi	Adresi	sehir	Telefon	talcep
1	Acarkan Eczanesi	Kordon Boyu No:7	Girne	815 33 75	
2	Akdeniz Eczanesi	Atatürk Cad.	Lefkoşa	228 82 77	
3	Akpınar Eczanesi	İnönü Bulvarı 45 C Sakarya	Gazimağusa	365 36 18	
4	Apaydın Eczanesi	2.Selim Cad. Arabacıoğlu Apt.	Lefkoşa	227 51 59	
5	Asar Eczanesi	K 40 Ecevit Cad.	Güzelyurt	714 21 93	
6	Ataçağ Eczanesi	Gönyeli	Lefkoşa	223 19 97	
7	Atakan Eczanesi	15 Ağustos Bulvarı Asımoğlu Pasajı	Gazimağusa	366 46 34	
8	Atun Eczanesi	İstiklal Cad.No:24	Gazimağusa	366 53 22	
9	Aycan Eczanesi	Ecevit Cad. No:4/G	Güzelyurt	714 21 93	
10	Ayşe Kaptan Eczanesi	Girne Cad.No:50	Lefkoşa	227 45 96	
11	Barış Eczanesi	Lapta	Girne	821 84 10	
12	Başak Eczanesi	Hürriyet Cad.No;95/A	Girne	815 36 20	
13	Bedeoğlu Eczanesi	Akdoğan	Gazimağusa	377 88 88	
14	Beril Eczanesi	Müftü Ziya Efendi Sok. Bekiroğlu İşhanı 28 B	Lefkoşa	227 73 10	
15	Bilge Eczanesi	Larnaka Yolu Özergin Apt.No:2	Gazimağusa	366 13 98	
16	Birgün Eczanesi	36/1 Cemal Aksay Cad. Kızılbaş	Lefkoşa	225 24 12	
17	Caglan Eczanesi	15 Ağustos Bulvarı No:34	Gazimağusa	366 64 47	
18	Ceren Eczanesi	15 Ağustos Bulvarı No:11/C	Gazimağusa	366 33 74	
19	Cevher Eczanesi	Dereboyu Kumsal	Lefkoşa	227 72 51	
20	Çakıcı Eczanesi	Mehmetçik Köyü	Gazimağusa	375 53 66	
21	Defne Eczanesi	Atatürk Cad.No:20	Girne	815 35 16	
22	Derman Eczanesi	Girne Cad. No:64	Lefkoşa	227 22 53	
23	Derya Eczanesi	Mehmet Akif Cad. No:53/3 Dereboyu	Lefkoşa	227 61 98	

with this table I finished my database. Now they are usable for converting Internet. After this I will start to use ASP and HTML codes. For writing these codes I must select a writing program. I chose the again Microsoft's product which is Frontpage. I chose it because for HTML it supports visual design and when I use ASP codes it helps for knowing these codes either HTML or ASP. When you start to write ASP codes, it changes these codes colors.(Fig.1.8.)

Fig.1.8.

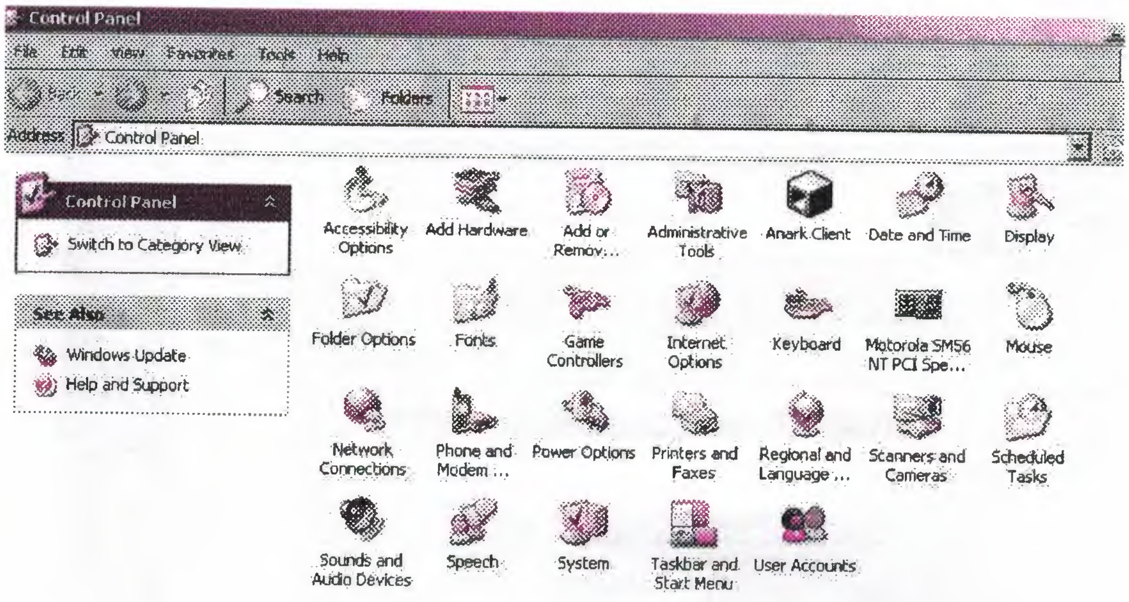

Microsoft FrontPage - F:\Inetpub\wwwroot\doktorlar20.asp
Dosya Düzen Görünüm Ekle Biçim Araçlar Tablo Çerçeveler Pencere Yardım
Yardım için soru yazın
doktorlar20.asp
Normal HTML Önizleme

```
<SCRIPT>
<!--
function popup(a,b,c) {
 var yeni;
 var en = (b < 300) ? 570 : ((b > 600) ? 600 : b);
 var boy = (c < 300) ? 270 : ((c > 980) ? 980 : c);
 yeni=window.open("","outputWindow","menubar=yes,scrollbars=yes,resizeble=yes,width="+en+",height="+boy);
 yeni.location=a;
 yeni.focus();
}
-->
</SCRIPT>

<body bgcolor="#C0C0C0" text="#000000" bgproperties="fixed" background="Image1.jpg">
<!--#include virtual="anketoy.asp"-->
<%
set conn=server.createobject("ADODB.connection")
conn.open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &Server.MapPath("doktor.mdb")
%>
<table width="193" border="0" align="center" cellpadding="2">
<tr>
<td bgcolor="#6666FF"><font size="1" face="tahoma" color="#FFFFFF"><b>Doktorlar</b></font></td>
</tr>
<tr>
```

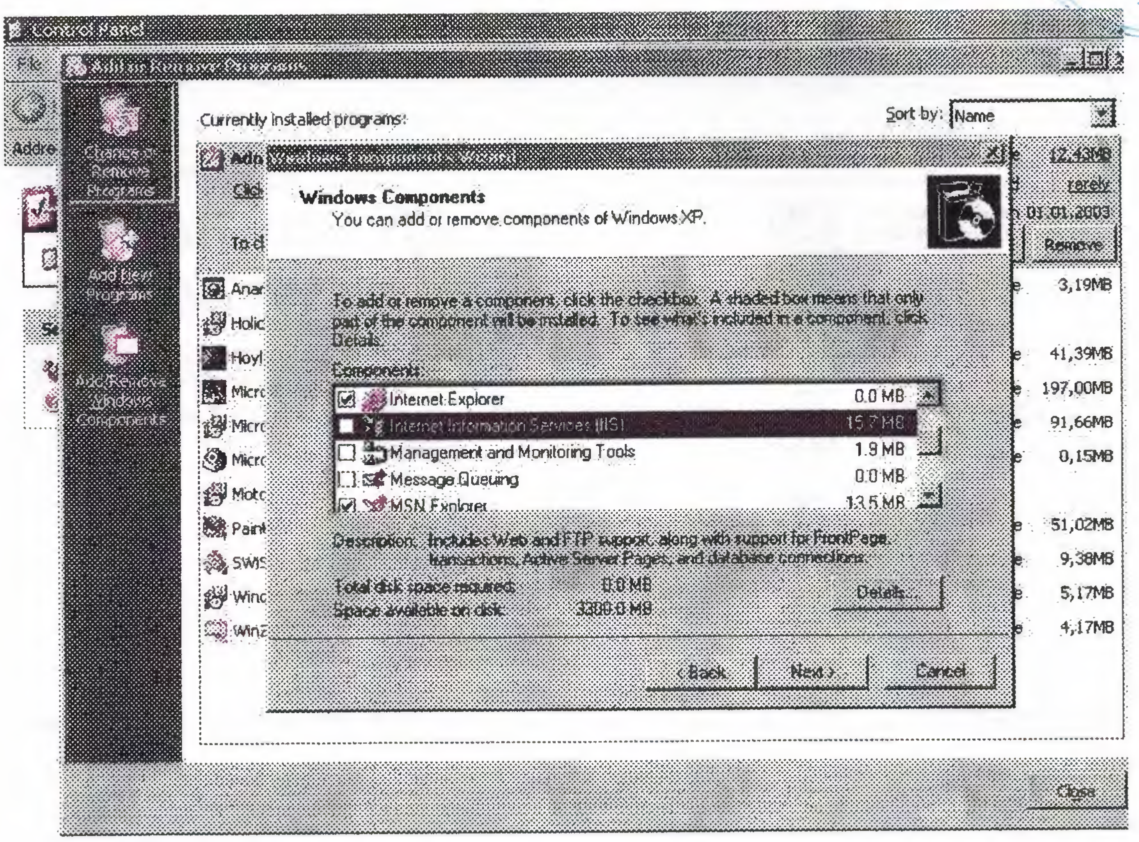
Up to here everything is clear I think. Before I start to write my ASP codes, I must make some changes in my computer. Because default windows doesn't support to your computer as server. Before I publish my web pages, I must see what I did in my computer, so I need a server and Microsoft gives a program for working computer as a server. For windows 98 this program is PWS(Personal Web Server). I used it before so I know. But today my operating system is win XP. With windows XP CD the IIS 5.1 coming but in default it is not on the computer. So we must set up. For setting up, after inserting win XP CD we come to start menu and select control panel.(Fig.1.9.)

Fig.1.9.



from this control panel I selected Add or Remove Programs. When I selected this from Add or Remove Programs menu, left side there was Add/Remove windows components. I pressed on that button.(Fig.1.10.)

Fig.1.10.



Currently installed programs:

Windows Components
You can add or remove components of Windows XP.

To add or remove a component, click the checkbox. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Components:

```
[✓] Internet Explorer                           0.0 MB
[ ] Internet Information Services (IIS)         15.7 MB
[ ] Management and Monitoring Tools              1.9 MB
[ ] Message Queuing                              0.0 MB
[✓] MSN Explorer                                13.5 MB
```

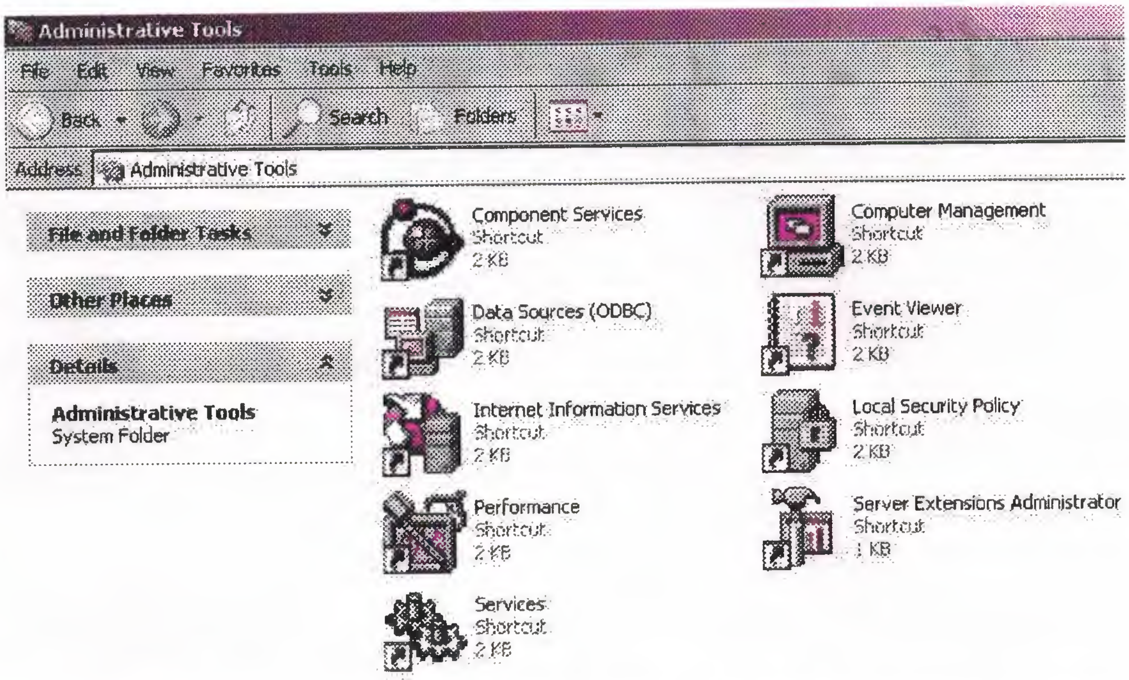
Description: Includes Web and FTP support, along with support for FrontPage transactions, Active Server Pages, and database connections.

Total disk space required: 0.0 MB
Space available on disk: 3300.0 MB

Details... < Back Next > Cancel

And from new menu I put tick for Internet Information Services(IIS) Then I pressed next. It automatically download the IIS 5.1. For seeing the job well done, we come to control panel(Fig.1.9.) and from this I selected Administrative Tools. Here, after I installed IIS I must see the internet information services symbol.(Fig.1.11.)

Fig.1.11

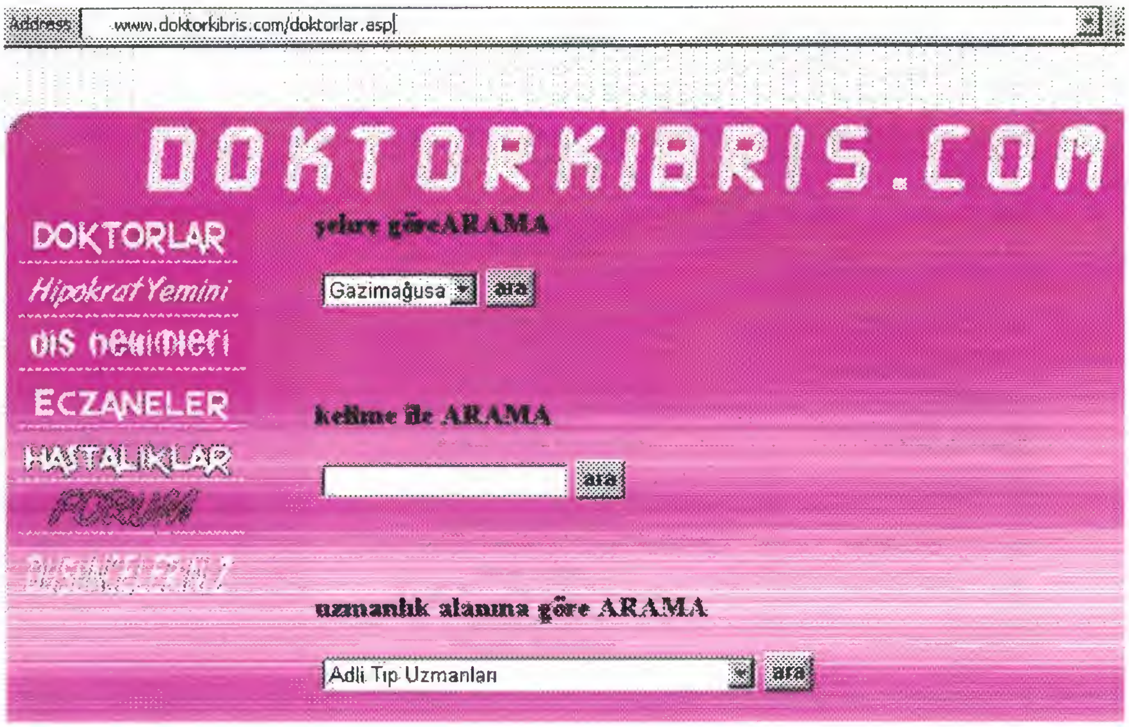


when I see this, it means that now I can use my computer as a server. However for using my computer as a server, I must save my asp or html files into the wwwroot folder which is under the our windows partition(generally c:/inetpub/wwwroot) if I don't save my files into that folder, I cant use server. This is very important point. For checking is it working, with IIS there was a file which name is iisstart.asp. when I write it into my browser if it is working, our server is working correctly. But here one more point, for seeing my web site, I must write address under this address: http://localhost/.........(e.g. http://localhost/iisstart.asp)

Till now, I just make preparation for ASP. For using ASP I must do that one time, after that I can write any codes with ASP. I want to show you my first page which is default page(default.htm)

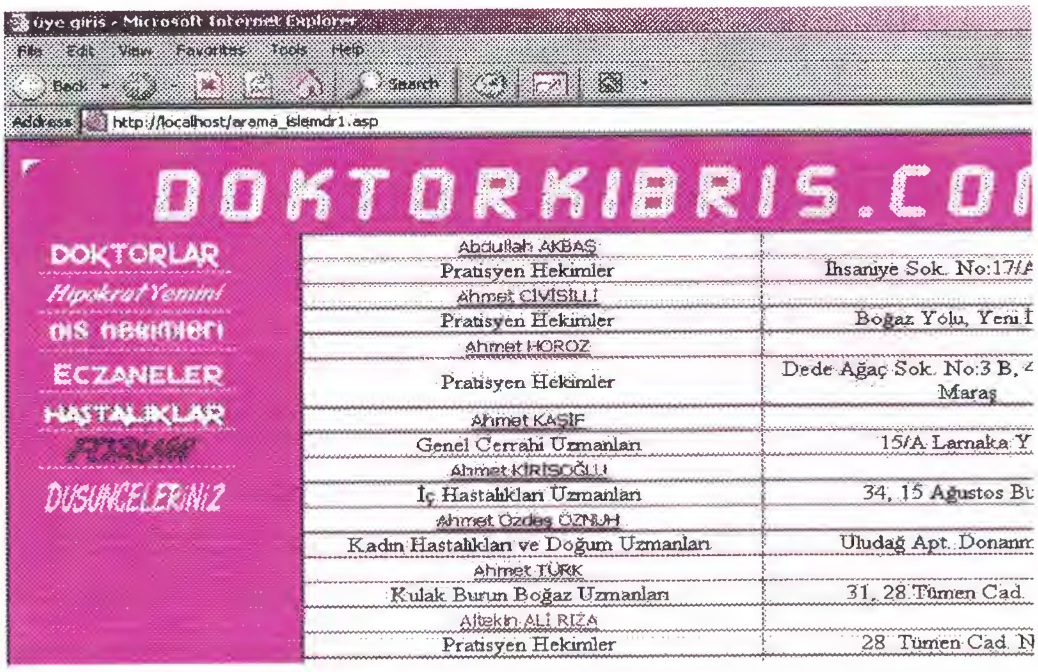
This is my default.htm page. Until here I didn't use any asp codes.

But after this I started to use them. In this page if you press DOKTORLAR button, you the new page will be opened, which name is (doktorlar.asp)



In this page you can search doctors by name, by surname, by city, by proficiency, and any keyword. I used here 3 type of search. Searching by cities(şehre gore arama) and Search by Proficiency(Uzmanlik alanına gore arama) took its data from directly Access Database. It means that if I add anything to my database, for example new proficiency or any city, I don't need to change my asp files. Because it will change automatically. Now we start to see ASP pages' activities.

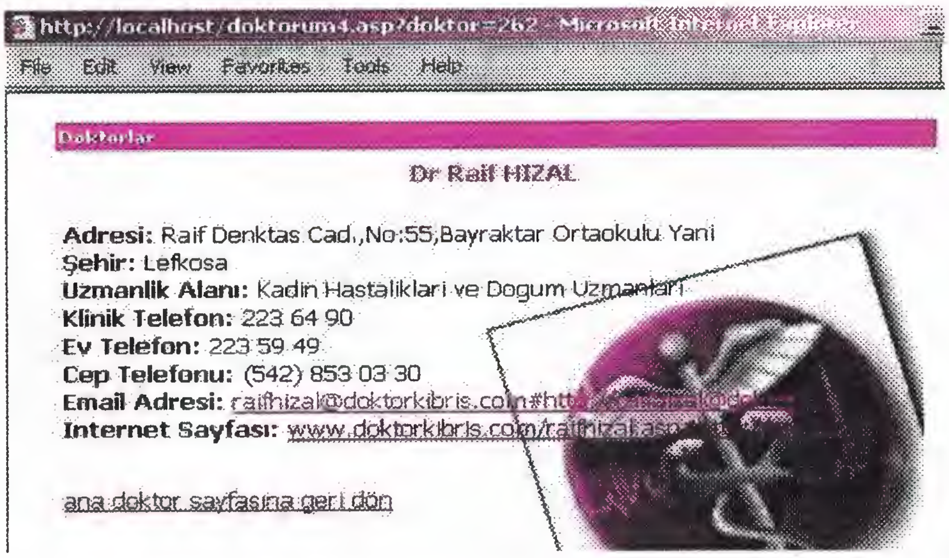
When we make search by city it will directly open a new page and it will give us results in "arama_islemdr1.asp" page. Here we can see results and for detailed results we can press the name of the Doctor.


üye giris - Microsoft Internet Explorer

Address http://localhost/arama_islemdr1.asp

Abdullah AKBAŞ	
Pratisyen Hekimler	İhsaniye Sok. No:17/A
Ahmet CIVISILLI	
Pratisyen Hekimler	Boğaz Yolu, Yeni İ
Ahmet HOROZ	
Pratisyen Hekimler	Dede Ağaç Sok. No:3 B, 4 Maraş
Ahmet KAŞİF	
Genel Cerrahi Uzmanları	15/A Larnaka Y
Ahmet KIRISOĞLU	
İç Hastalıkları Uzmanları	34, 15 Ağustos Bu
Ahmet Özdeş ÖZNUH	
Kadın Hastalıkları ve Doğum Uzmanları	Uludağ Apt. Donanm
Ahmet TÜRK	
Kulak Burun Boğaz Uzmanları	31, 28 Tümen Cad.
Altekin ALİ RIZA	
Pratisyen Hekimler	28 Tümen Cad. N

When we press any name, it will open new page and it will show us that doctor's details, which are not only Name and Address but also city, telefon number for clinic and gsm number. If that doctor has a web site or email, we can see them directly. That pages name is "doktorum4.asp".


http://localhost/doktorum4.asp?doktor=262 - Microsoft Internet Explorer

Doktorlar

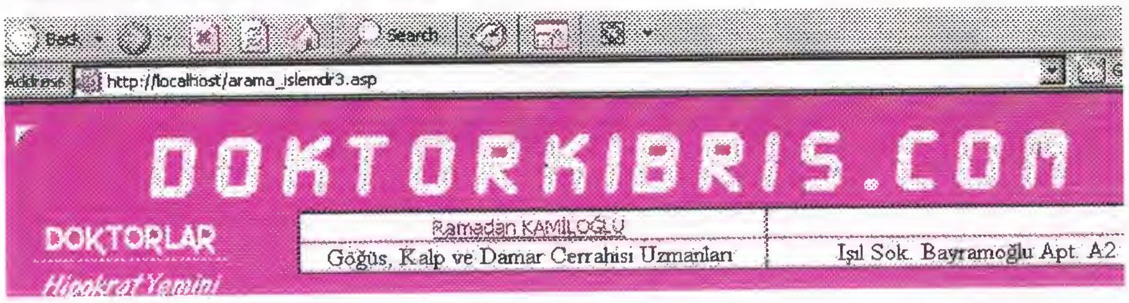
Dr Raif HIZAL

Adresi: Raif Denktas Cad.,No:55,Bayraktar Ortaokulu Yani
Şehir: Lefkosa
Uzmanlik Alanı: Kadin Hastaliklari ve Dogum Uzmanlari
Klinik Telefon: 223 64 90
Ev Telefon: 223 59 49
Cep Telefonu: (542) 853 03 30
Email Adresi: raifhizal@doktorkibris.com#htt[illegible]
Internet Sayfası: www.doktorkibris.com/raifhizal.asp

ana doktor sayfasina geri dön

If we search by a keyword, our pages start to search from database, and it will check all cities, names, address, proficiency. If it find anything like search criteria, it will give us results in "arama_islemdr2.asp" page.

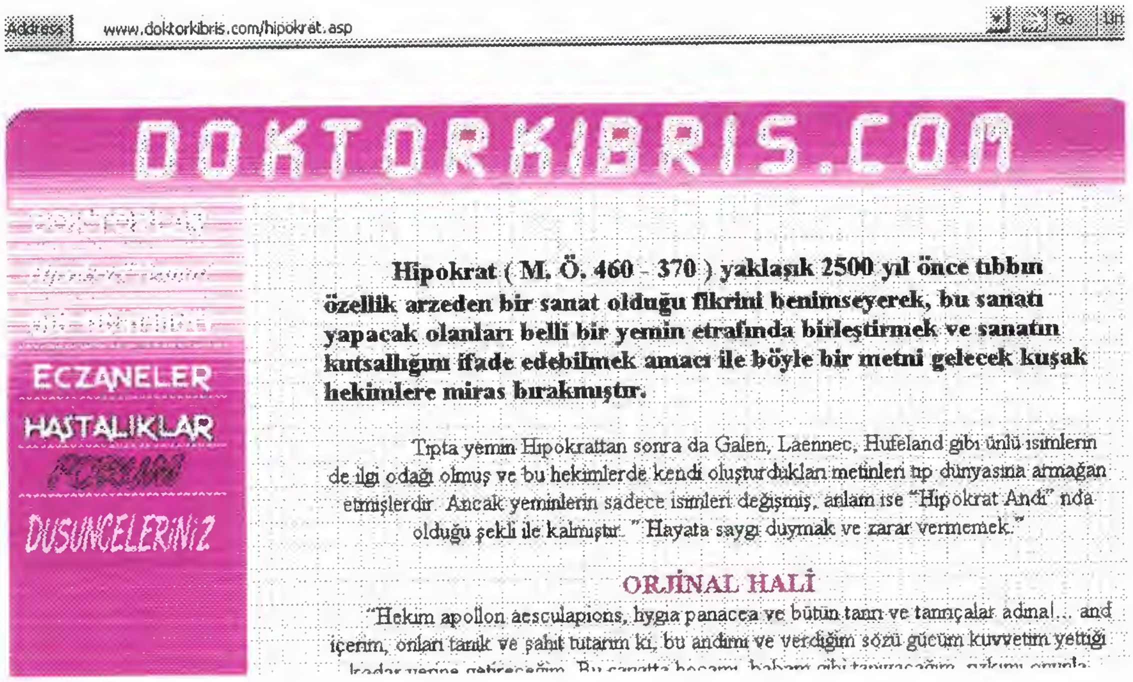
Doktorkibris.com has one more search type which is by proficiency. If you are searching a doctor for your heart you can select one of them from my page results. It will lists all of them in "arama_islemdr3.asp" page.



Ramadan KAMİLOĞLU	
Göğüs, Kalp ve Damar Cerrahisi Uzmanları	Işıl Sok. Bayramoğlu Apt. A2

In my main menu under the "DOKTORLAR" there is interesting point which name is known by everybody but until now I didn't know its original part. I searched it in internet but I found its Turkish version which must be sweared by all doctors. Hipokrat Sweared is in "hipokrat.asp" page.

Fig.1.



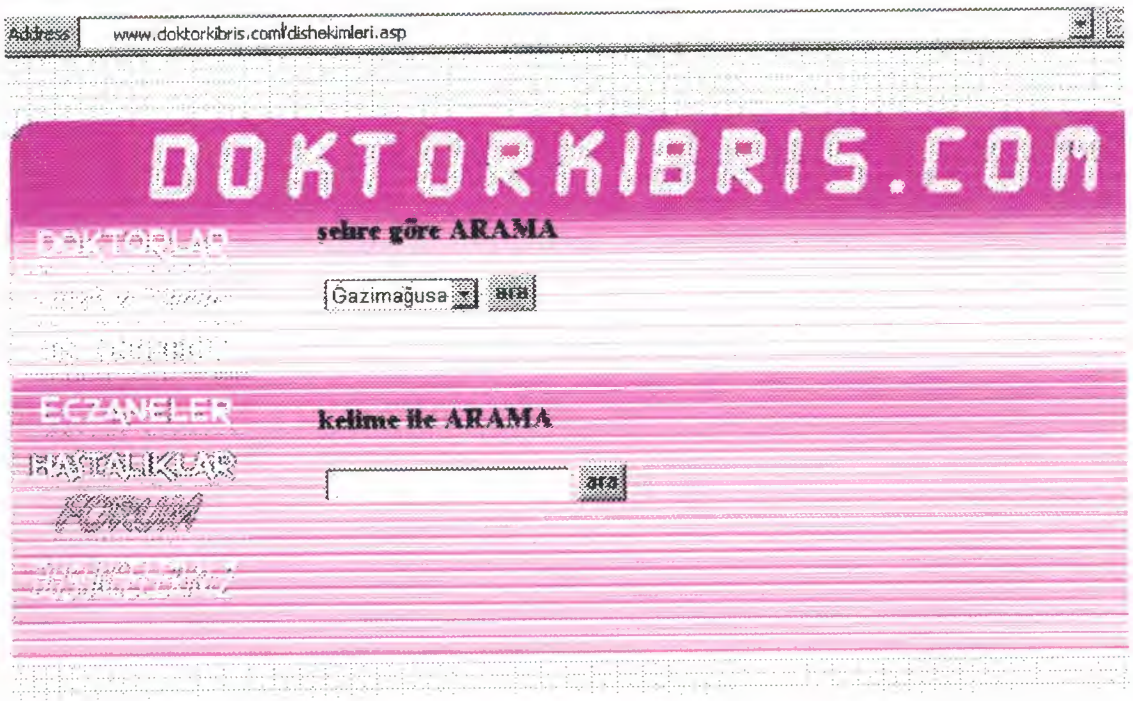
Hipokrat (M. Ö. 460 - 370) yaklaşık 2500 yıl önce tıbbın özellik arzeden bir sanat olduğu fikrini benimseyerek, bu sanatı yapacak olanları belli bir yemin etrafında birleştirmek ve sanatın kutsallığını ifade edebilmek amacı ile böyle bir metni gelecek kuşak hekimlere miras bırakmıştır.

Tıpta yemin Hipokrattan sonra da Galen, Laennec, Hufeland gibi ünlü isimlerin de ilgi odağı olmuş ve bu hekimlerde kendi oluşturdukları metinleri tıp dünyasına armağan etmişlerdir. Ancak yeminlerin sadece isimleri değişmiş, anlam ise "Hipokrat Andı" nda olduğu şekli ile kalmıştır. " Hayata saygı duymak ve zarar vermemek."

ORJİNAL HALİ

"Hekim apollon aesculapions, hygia panacea ve bütün tanrı ve tanrıçalar adına!... and içerim, onları tanık ve şahit tutarım ki, bu andımı ve verdiğim sözü gücüm kuvvetim yettiği

Another search pages are for dentist search (Diş Hekimleri). In this page I used two types of searching because under the dentists there is no root as doctors. You can search by keyword("arama_dislemdr2.asp") or by city("arama_dislemdr1.asp"). from these results we can see details of any dentist by just pressing on the name. It will open "disciler4.asp" page.



Preparing medical web page, in my opinion cannot be without pharmacies. Therefore I collected all data about pharmacies and I prepared search page for all pharmacies in Cyprus.“ eczaneler.asp”. Here we can search by keyword or by city. When we search by keyword “arama_islemeczane2.asp” pages will show you results. For searching by city “arama_islemeczane.asp” page will help you.

Another topic is in my webpage, disease(hastalıklar). You can see lots of diseases and you can read about these diseases articles. This page can be uploaded any time I want.