Dr. Abdallah Al-Sukairi

sukairi@kfupm.edu.sa

Information & Computer Science Department

Web / Database Integration

Outline

- Introduction
- Building Active Server Pages
- Working with HTML Forms
- ASP Installable Components
- Using ASP with Databases

Web-DBMS Architecture

- Three-tier architecture
  - First Tier (Client)
    - a Web browser acting as a 'thin' client
  - Second Tier (Application Server)
    - a Web server
  - Third Tier (Database Server)

Approaches to the Web-DBMS Integration

- Common Gateway Interface (CGI)
- Server-Side Includes
- Extension to the Web server
- Java and JDBC, JSQL, JRB
- Scripting languages such as JavaScript and VBScript
- Microsoft's Active Platform
- Oracle's Network Computing Architecture (NCA)

CGI (Common Gateway Interface)

- CGI is a specification to allow a program (the CGI program) to control the information passed between a Web browser and a Web server during a connection session
- The CGI program executes at the Web server
- The output from the CGI program is passed back by the Web server to the Web Browser
- Information from the browser is passed to the CGI program via environment variables or read from standard input
- A URL can point to a CGI program
- Form's data is passed to the CGI program specified in the Action parameter

Active Server Pages (ASP)

- Microsoft ASP is a programming model that allows dynamic, interactive Web pages to be created on the Server
- VBScript is the default scripting language for ASP
- ASP files can contain: text, HTML tags, Script command
- When a browser request and '.asp' file, the Web server calls ASP, which reads the file and executes any commands, and sends the generated HTML page to the browser

. . . Active Server Pages (ASP)

- **With ASP we can**
  - Generate dynamic Web pages
  - Process content of HTML forms
  - Create database-driven Web pages
  - Track user sessions
  - Create searchable Web pages
  - Detect the capabilities of different browsers
  - Send and retrieve e-mail
  - Integrate custom components into Web site
- **Internet resources**
  - aspsite.com, activeserverpages.com, 15seconds.com
    serverobjects.com, asphole.com
    msdn.microsoft.com/workshop/server/default.asp
    microsoft.com/ntserver/web/

Building Active Server Pages

How ASP Works

- IIS
- Static HTML pages
- .asp dynamic pages
- ASP.dll
- A server-side scripting environment
- ASP editor
- Script delimiters <% and %>
- Script language
  - IIS Default ASP language
  - <% @ Language=JScript %>
  - <script language="JScript" runat="server">

Examples

```
<body>
This is a
<% for i=1 to 10 %>
very,
<% next %>
very long sentence.
```

```
<% If Time >= #12:00:00 AM# And Time < #12:00:00 PM#
Then %> Good Morning!
<% Else %>
Hello!
<% End If %>
```

Output Directives

- `<% = ? %>`
  - `<% =Time %>`
- **Response.Write() method**
  - **<% Response.Write(Time) %>**
- **Example**
  - ```
    <body>
    <%
    for i=1 to 10
    var = var & "very, "
    response.write(i & ": " & var & "<br>")
    next
    %>
    ```
    ```

Objects and Components

- **ASP has several built-in objects**
  - Each object has: methods, properties, and possibly collections (key and value pairs)
  - Application, Request, Response, Server, Session, ObjectContext
  - No need to create an instance of a built-in object
- **ASP Components**
  - Extend to the power of ASP
  - Third-party and and user created ActiveX components
  - Used in specialized tasks
  - Some of the components bundled with ASP
    - Ad Rotator, Browser Capabilities, Content Linking, Counters, Content Rotator, Page Counter, Permission Checker, Collaboration Data Objects (CDO), ActiveX Data Objects (ADO)

Hypertext Transfer Protocol (HTTP)

- A request and response protocol
- A browser initiates a request
  - Method of request + Header
  - Get /hello.htm HTTP/1.1
Host: www.aspsite.com
- HTTP response
  - status line + header(s) + possibly a message body
- A built-in ASP Response object
- Request object

Using the `Response` Object

- **Response.Write**
- **The output from an ASP page is sent to the browser immediately after each command is executed**
  - ```
    <body>
    <%
    for i=1 to 500
      response.write(i & "<br>")
    next
    %>
    ```
    ```

Controlling ASP Caching

- Browser and Proxy caching to improve performance
- The Response object has three properties to control page caching
  - Browser caching: Expires, ExpiersAbsolute,
  - Proxy server caching: CacheControl

```
<body>
The time is <% =Time() %>
</body>
```

. . . Controlling ASP Caching

```
<% Response.Expires = 0 %>


<% Response.ExpiresAbsolute = #JAN 2, 2000 00:00:00# %>
<html>
<head> <title>The Time</title> </head>
<body>
The Time is: <%=TIME()%>
</body>
</html>


 <% Response.CacheControl= "Private" %>    DEFAULT


<% Response.CacheControl= "Public" %>
```

Using the `Request` Object

- **Three collections**
  - `QueryString, Form, ServerVariables`
- **The `QueryString` collection**
  - Part of a page request that appears after the ?
  - http://www.aspsite.com/hello.asp?username=Ahmed+Mustafa
  - Usually used to pass information from one ASP to another

```
<HTML>
<HEAD> <TITLE> Query String </TITLE> </HEAD>
<BODY>
Welcome <%=Request.QueryStringm("username")%>!
</BODY>
</HTML>
```

The `ServerVariables` Collection

- When a browser requests a Web page, the request includes several headers

```
<HTML>
<HEAD><TITLE>Server Variables</TITLE></HEAD>
<BODY>
<%
FOR EACH name IN Request.ServerVariables
  Response.write("<P><B>"&name&"</B>:")
  Response.write(Request.ServerVariables(name))
NEXT
%>
</BODY>
</HTML>
```

Server Variables

APPL_MD_PATH:/LM/W3SVC/1/ROOT
APPL_PHYSICAL_PATH:C:\WEBSHARE\WWWROOT\
AUTH_PASSWORD:
AUTH_TYPE:
AUTH_USER:
CONTENT_LENGTH:0
CONTENT_TYPE:
GATEWAY_INTERFACE:CGI/1.1
INSTANCE_ID:1
INSTANCE_META_PATH:/LM/W3SVC/1
LOCAL_ADDR:196.1.65.160
LOGON_USER:
PATH_INFO:/test.asp
PATH_TRANSLATED:C:\WEBSHARE\WWWROOT\test.asp
QUERY_STRING:
REMOTE_ADDR:196.1.65.160
REMOTE_HOST:196.1.65.160
REMOTE_USER:
REQUEST_METHOD:GET
SCRIPT_NAME:/test.asp
SERVER_NAME:ics-sukairi
SERVER_PORT:80
SERVER_PORT_SECURE:0
SERVER_PROTOCOL:HTTP/1.1
SERVER_SOFTWARE:Microsoft-IIS/5.0
URL:/test.asp
HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
HTTP_ACCEPT_LANGUAGE:en-us,ar;q=0.5
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:ics-sukairi
HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
HTTP_COOKIE:ASPSESSIONIDGQQGGMBK=GDDLNJGBECACBBFIDLGPFLMN
HTTP_ACCEPT_ENCODING:gzip, deflate

Working with HTML Forms

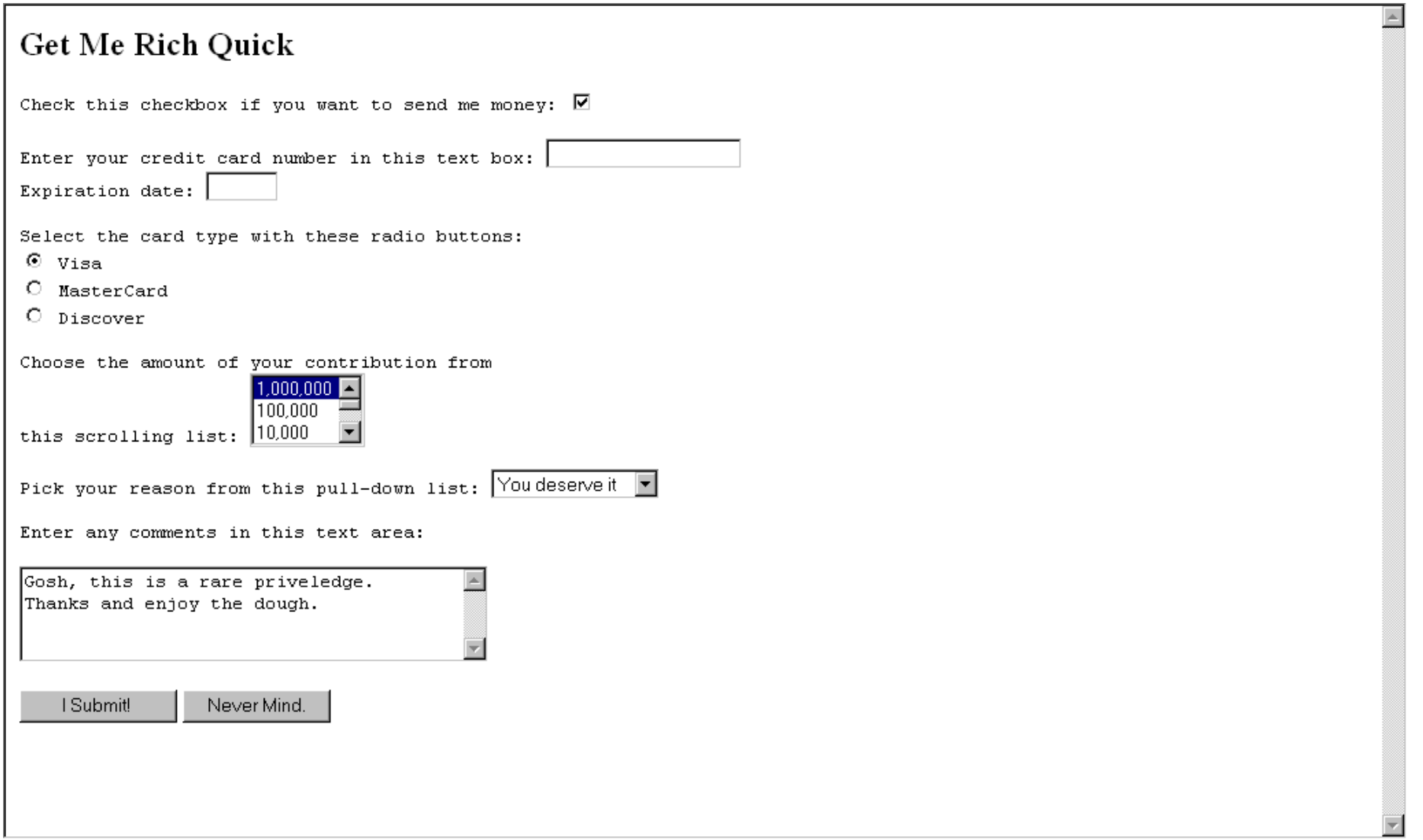
HTML Forms

- Interactive Web pages
- Client-side processing
- Server-side processing
- following tags are used to create forms
  - `<FORM>` A form within a document
    `<INPUT …>` One input field
    `<OPTION>` One option within a Select element
    `<SELECT>` A selection from a finite set of options
    `<TEXTARAE …>` A multi-line input field
    `<LABEL …>` Active Control Labels
  - Each variable field is defined by an `INPUT`, `TEXTAREA`, or `OPTION` tags and must have a `NAME` attribute

Creating a Form

- `<FORM ACTION="address of a script to process this form" METHOD="GET|POST" ENCTYPE="MIME type">`
  `. . .`
  `</FORM>`
- `ACTION`
  - URL of a script or application
  - `ACTION="mailto:order@company.com"`
- `METHOD`
  - The `GET` attribute is ideal for form submission
  - The `POST` attribute is ideal for forms used to provide information
- `ENCTYPE`
  - Specifies the media type used to encode the form data
  - The default ENCTYPE is the MIME type 'application/x-www-form-urlencoded'
    - Spaces are replaces by "+"
    - Non-alphanumeric characters are replaces by "%nn" (nn is the ASCII code)
    - text/plain type
- Forms can not be nested

Form Examples



… Form Examples

```
<BODY>
<H2>Get Me Rich Quick</H2>
<FORM action= method=post>
<INPUT name=cant_see_me type=hidden value="the user won't see this">
<PRE> Check this checkbox if you want to send me money:
<INPUT name=checkme type=checkbox CHECKED>
Enter your credit card number in this text box: <INPUT name=cardnum>

Expiration date: <INPUT name=ext size=5>

Select the card type with these radio buttons:
<INPUT name=payment type=radio CHECKED value=v> Visa
<INPUT name=payment type=radio value=m> MasterCard
<INPUT name=payment type=radio value=d> Discover

Choose the amount of your contribution from this scrolling list:
<SELECT name=howmuch size=3>
<OPTION selected>1,000,000 <OPTION>100,000 <OPTION>10,000<OPTION> 1,000
<OPTION>100 <OPTION>Whatever </OPTION> </SELECT>

Pick your reason from this pull-down list:
<SELECT name=why> <OPTION selected>You deserve it<OPTION>I don't deserve it
<OPTION>I'm rich<OPTION>I'm stupid<OPTION>I love you<OPTION>What the heck
</OPTION></SELECT>

Enter any comments in this text area:<BR>
<TEXTAREA cols=40 name=comments rows=4>Gosh, this is a rare priveledge.
Thanks and enjoy the dough.</TEXTAREA> </PRE>

<INPUT type=submit value="    I Submit!     ">
<INPUT type=reset value="Never Mind.">
</FORM></BODY>
```

Retrieving Form Data

```
<HTML>
<HEAD><TITLE> Simple Form </TITLE></HEAD>
<BODY>
<FORM METHOD="post" action="result.asp">
<P>Please Enter Your User Name:
<BR><INPUT NAME="username" TYPE="text">
<P>Please Enter Some Comments:
<BR><TEXTAREA NAME="comments" COLS=40 ROWS=5></TEXTAREA>
<P><INPUT TYPE="submit"VALUE="Save">
</FORM>
<BODY>
</HTML>
```

… Retrieving Form Data

- The `Request.Form` collection
- `Result.asp`

```
<%
DIM username, comments
username = Request.Form("username")
comments = Request.Form("comments")
%>
<HEAD><TITLE> Result </TITLE></HEAD>
<BODY>
Your User Name Is: <%=username%>
<P>
Your Comments Are: <%=comments%>
<BODY>
</HTML>
```

Validating Form Data

- Client-side validation
- Server-side validation

```
<html>
<head><title>Simple HTML Form</title></head>
<body>
<form method="post" action="result.asp">
<b>Please enter your name: </b>
<input name="username" type="text" size=30><p>
<b>Please enter your date of birth: </b>
<input name="birthdate" type="text" size=10><p>
<input type="submit" value="Save">
</form></body></html>
```

```
<html>
<head><title>Result</title></head>
<body>
Thank you for registering!
</body>
</html>
```

ASP Installable Components

Page Counter Component

- **Three methods**
  - `Hits(path)`
    - Displays the number of times that a specified URL has been opened
  - `PageHit()`
    - Increments the Hit Count
  - `Reset(path)`
    - Sets the hit count for a specified page to 0

```
<HTML>
<HEAD><TITLE> Page Counter Example </TITLE></HEAD>
<BODY>
<%
Set MyHits=Server.CreateObject("MSWC.PageCounter")
MyHits.PageHit()
%>
This page has been viewed <%=MyHits.Hits%> times.
</BODY></HTML>
```

Counter Component

- Can be used to count many things
- Counter object in global.asa
  - `<OBJECT RUNAT=Server SCOPE=Application ID=Counter PROGID="MSWC.Counters"> </OBJECT>`
- Methods
  - Get
    - Returns the value of the counter
  - Increment
    - Increases the counter by 1
  - Remove
    - Removes the counter from the Counters.txt file
  - Set
    - Sets the value of the counter to a specific integer

Example

```
<%
theBrowser = UCASE( Request.ServerVariables( "HTTP_USER_AGENT" ) )
if instr( theBrowser, "MSIE" ) > 0 THEN
  MyCount.Increment( "Microsoft" )
else
  MyCount.Increment( "Netscape" )
end if
%>

<HTML>
<HEAD><TITLE>Some Page</TITLE></HEAD>
<BODY>
<p><b>Microsoft Visitors:</b><%=MyCount.Get( "Microsoft" ) %>
<p><b>Netscape Visitors:</b><%=MyCount.Get( "Netscape" )%>
</BODY>
</HTML>
```

Displaying Advertisement

- The `Ad Rotator` component creates an Ad Rotator object that automates the rotation of advertisement images on a Web page
  - `Set AdRotator = Server.CreateObject( "MSWC.AdRotator" )`
- Properties
  - Boarder
    - Specifies the size of the border around the advertisement
  - Clickable
    - Specifies whether the advertisement is a hyperlink
  - TagetFrame
    - Specifies the name of the frame in which to display the advertisement
- Methods
  - GetAdvertisement
    - Gets the specifications for the next scheduled advertisement from the data file and formats it as HTML

Example

```
<HTML>
<HEAD><TITLE> Home Page </TITLE></HEAD>
<BODY>
<CENTER><H1>Welcome to our web site!</H1></CENTER>
<HR>
<%
Set MyAd=Server.CreateObject("MSWC.AdRotator")
%>
<CENTER><%= MyAd.GetAdvertisement("adrot.txt") %></CENTER>
</BODY>
</HTML>
```

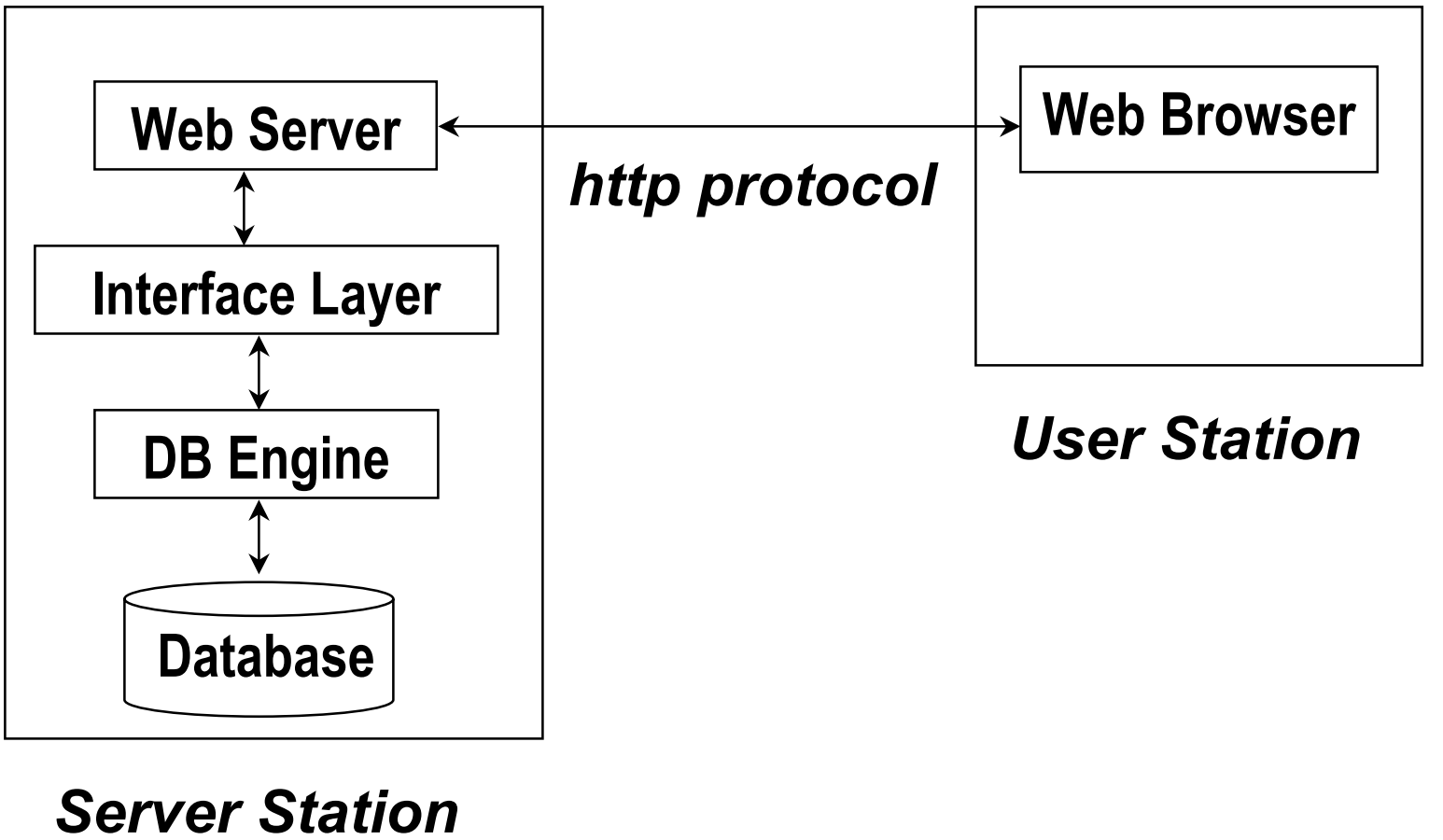
The Rotator Schedule File

- A text file that contains the display schedule and file information for advertisements. This file must be available on a Web server virtual path
- Redirection File
  - An optional file that implements redirection and enables the Ad Rotator component to record how many users click on each advertisement

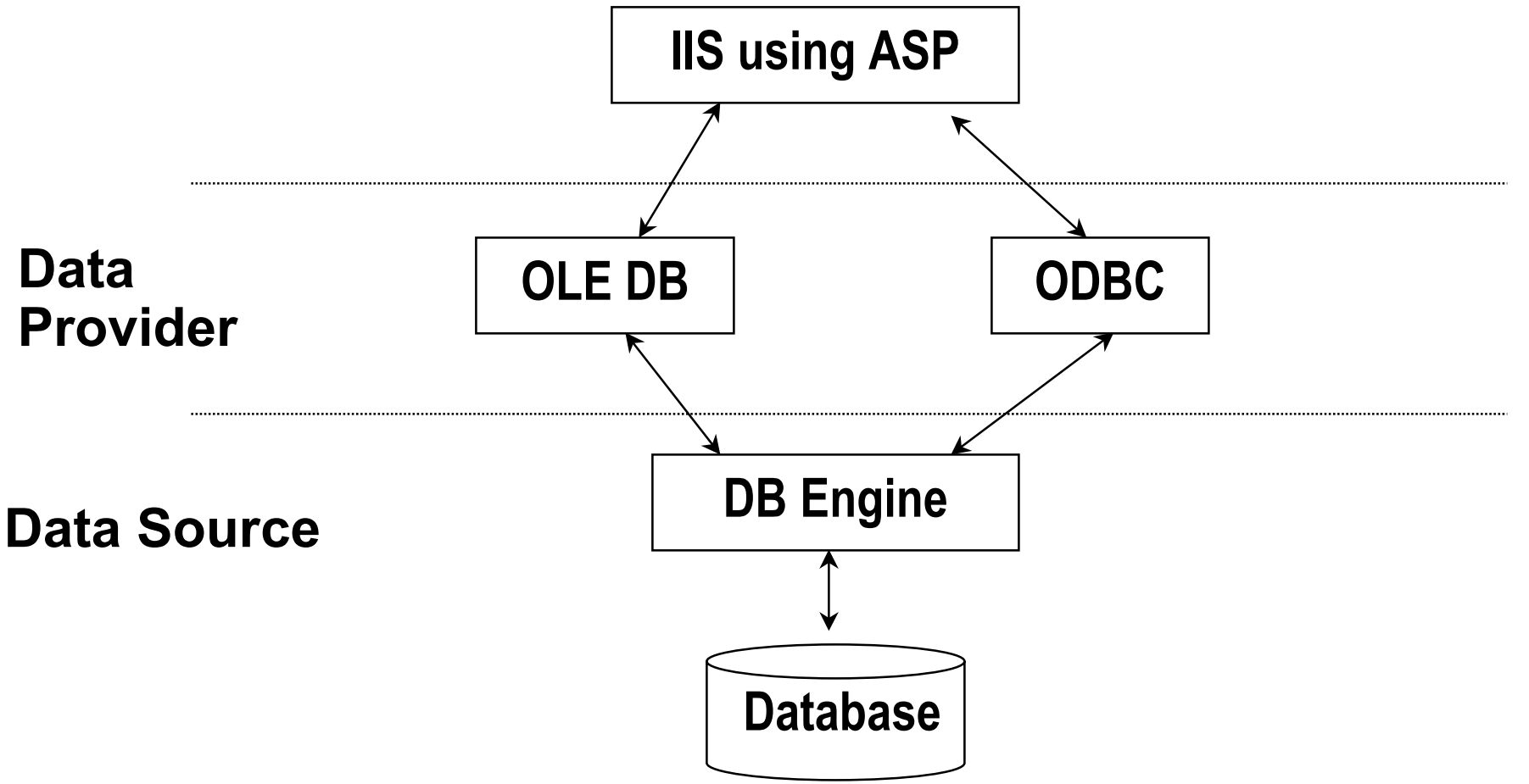
```
REDIRECT adredir.asp
WIDTH 468
HEIGHT 60
BORDER 0
*
aspsitead.gif
http://www.aspsite.com
The Active Server Pages Site
80
cityauctionad.gif
http://www.cityauction.com
CityAuction
20
```

Using ASP with Databases

Generic Model for Web-Database Access



Microsoft's Solution



… Microsoft's Solution

- ASP uses ADO (ActiveX Data Objects) as the programming interface
- ADO uses either ODBC or OLE DB
- OLE DB is meant to replace ODBC
  - Native OLE DB driver is faster than ODBC
  - OLE DB drivers for many data sources (SQL Server, Oracle, Microsoft Jet, Text file, ADSI, Exchange)
  - There is a generic OLE DB driver for ODBC
- The Microsoft Data Access Components (MDAC)
  - Contain all components for database access
  - Downloadable from http://www.Microsoft.com/data

ADO Top Level Objects

- `Connection` **Object**
  - to establish a connection with a data source
  - execute queries
- `Recordset` **Object**
  - represents a set of records returned by a select query
- `Command` **Object**
  - to execute SQL Server stored procedures

```
<%
Set Con=Server.CreateObject("ADODB.Connection")
Response.Write "ADO Version = " & Con.Version
%>
```

Using the `Connection` Object

- **Create a `Connection` object**
  - `Set Con= Server.CreateObject("ADODB.Connection")`
- **Use the `Open` method and specify as a parameter either**
  - **an OLE DB connection string**

    ```
    Con.Open "Provider=SQLOLEDB; Data Source=myServer;
    UID=sa; PWD=secret; DATABASE=Pubs"
    ```

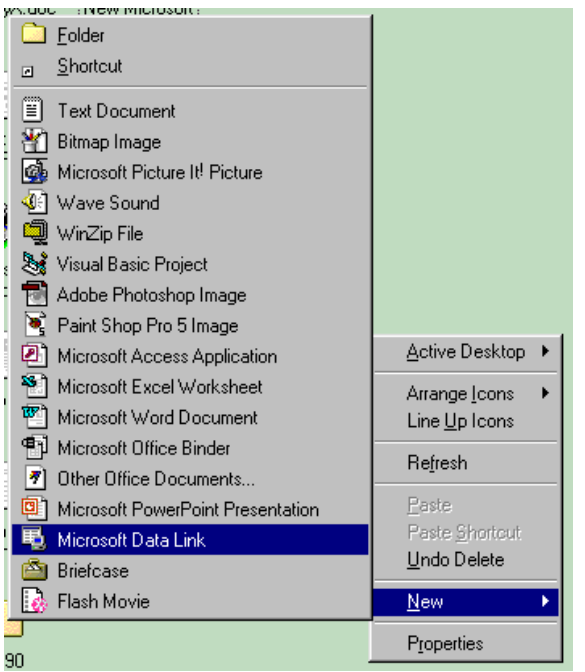
  - **an ODBC Data Source Name (DSN), or ODBC connection string**

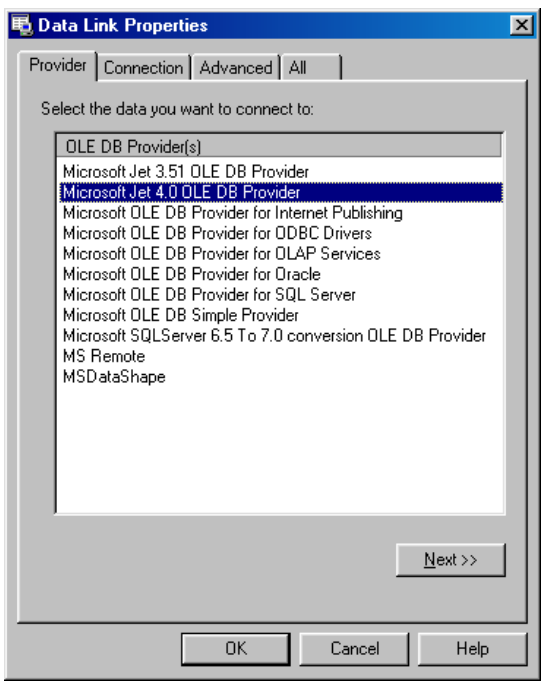
    ```
    Con.Open mydsn

    Con.Open "DSN=mydsn;UID=sa;PWD=secret;DATABASE=Pubs"
    ```

Connecting to a MS Access Database

- **Using OLE DB**
  - ```
    <%
    Set Con= Server.CreateObject("ADODB.Connection")
    Con.Open "Provider=Microsoft.Jet.OLEDB.4.0;
    Data Source=c:\myAccess.mdb"
    %>
    ```
  - ```
    Con.Open "File Name=c:\myDataLink.udl"
    ```





Using Connection Execute Method

- **To execute Insert/Delete/Update query**
  - ```
    St1 = "Insert into tUser
    values ('Ali', 'secret')"
    St2 = "Delete * from tUser"
    St3 = "Update tUser SET password=" "
    Con.Execute st1
    Con.Execute st2
    Con.Execute st3
    ```
    ```

Using Transactions

```
Con.BeginTrans
Con.Execute "Insert tUserX Select * from tUser"
Con.Execute "Delete * from tUser"
Con.CommitTrans
```

Transaction Support in ASP

- Use `BeginTrans/CommitTrans` of a Connection object
- Use MTS (Microsoft Transaction Server) and designate an ASP page as transactional
- Use `BeginTrans/CommitTrans` within a stored procedure in an SQL Server database and use the ADO Command object to execute the stored procedure

Using the Recordset Object

- **Use a recordset (rs) of a Select query**
  - `Set rs = Con.Execute ("select * from tUser")`
- **The `Recordset` object provide sophisticated control over how the result of a query is accessed and manipulated.**
- **Cursor control (`CursorType`): `adOpenForwardOnly`, `adOpenStatic`, `adOpenDynamic`, `adOpenKeyset`**
- **Locking control (`LockType`): `adLockReadOnly`, `adLockPessimistic`, `adLockOPtimistic`, `adLockBatchOptimistic`**

Example

```
<html>
<head><title>Show Table</title>
<%
Set Con = Server.CreateObject( "ADODB.Connection" )
Con.Open "FILE NAME=c:\myDataLink.UDL"
Set RS = Con.Execute( "select * from Authors ORDER BY au_lname" )
%>
<table border=1>
<% While NOT RS.EOF %>
  <tr>
    <td> <%=RS( "au_lname" )%> </td>
    <td> <%=RS( "au_fname" )%> </td>
    <td> <%=RS( "phone" )%> </td>
  </tr>
<%
RS.MoveNext
Wend
RS.Close
%>
</table>
</body>
</html>
```

Recordset Cursor Types

- `adOpenForwardOnly`
  - (Default) used for fastest performance
  - records are fetched serially from first to last (cannot move back)
- `adOpenStatic`
  - supports back/forward movement but cannot detect changes by other users
- `adOpenKeyset`
  - detect update changes but not insert/delete
- `adOpenDynamic`
  - (Richest) detects all changes

Recordset Lock Types

- `adLockReadOnly`
  - (Default) Allows read by multiple users
  - Data cannot be edited
- `adLockPessimistic`
  - Other users cannot access the record as soon as editing starts
- `adLockOptimisitic`
  - Other users can access the record but not at the moment changes are to be committed
- `adLockBatchOptimisitic`
  - used with in conjunction with UpdateBatch method

Opening A Recordset

- Create a recodset object and set the required options
- Open the recodset using a query and a predefined connection
- Example:

```
<!-- #Include virtual="/adovbs.inc" -->
<% Set Con= Server.CreateObject("ADODB.Connection")
Con.Open mydsn
Set  RS= Server.CreateObject("ADODB.Recordset")
RS.CursorType = adOpenStatic
RS.Open "Select * from Authors", Con
Response.Write "The recordset contains " &
   RS.RecordCount  & " Records"
```

Accessing Fields Collection in a Recordset

- The fields are indexed from 0 to RS.Fields.Count-1
- Fields is the default collection of a recodset and thus the syntax RS(I) and RS.Fields(I) are equivalent where I can be an index value or a field name
- To access the value of the first field, named phone, use RS(0) or RS.Fields(0) or RS("phone") or RS.Fields("Phone"), or RS(0).Value, ..
- To access the name of the field, use RS(0).Name

Scrolling Through a Recordset

- **Move**
  - moves forward or backward a number of records relative to the current record or a bookmark
- **MoveFirst (MoveLast)**
  - moves to the first (last) record
- **MoveNext (MovePrevious)**
  - moves to the next (previous) record
- **Use BOF (EOF) to detect the beginning (end) of a recordset**

Adding/Editing Records in a Recordset

- **To add a record open an adLockOptimistic Recordset and use AddNew/Update methods**
  - **RS.AddNew**
  - **RS("Name") = "Ali"**
  - **RS("Phone") = "8601234"**
  - **RS.Update**

Building Parameterized Queries

- Parameters are typically used in the where clause in a query
  - "Select * from tCar where color = ' " & colorval & " ' "
- A parameter value can be passed through form data or query string
  - colorval = Request.QueryString("color"), or
  - colorval = Request.Form("color")

Example 1

Cars for Sale - Microsoft Internet Explorer

File Address ars.asp Go Links

Cars for Sale

Maker - Model - Year	Price	Color
Ford Tempo 95	50000	paige
Toyota Sunny 97	32000	brown
Nissan Ultima 96	30000	grey
Nissan Cedric 95	30000	green
Toyota Cressida 96	30000	blue
Nissan Ultima 94	25000	silver

Internet

Database Query - Microsoft Internet Explorer

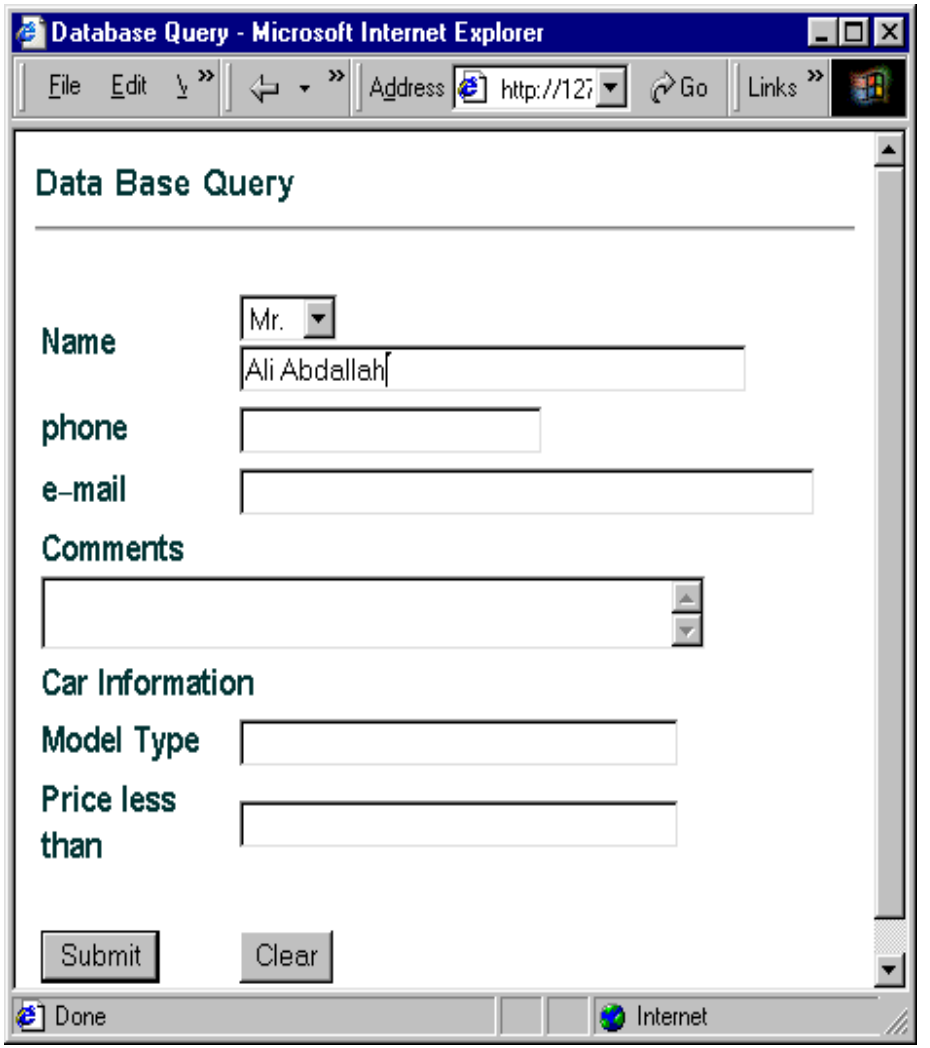
File Address http://127 Go Links

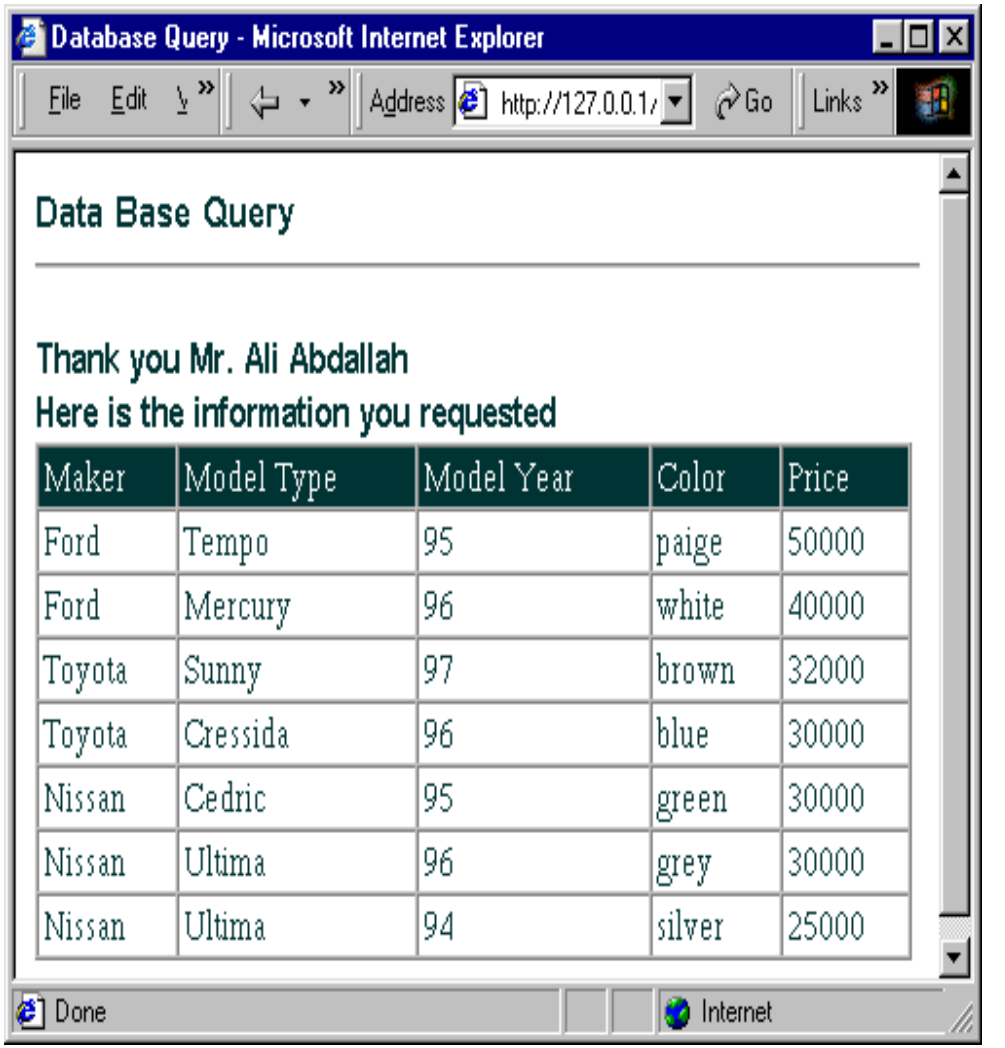
Car Information - Ford Tempo 95

Price	50000
Color	paige
Mileage (km)	600000
Contact Name	Sami Ihsan
Contact Phone	8602125

Done Internet

Example 2


Database Query - Microsoft Internet Explorer
File Edit
Address http://127
Go
Links
Data Base Query
Name
Mr.
Ali Abdallah
phone
e-mail
Comments
Car Information
Model Type
Price less than
Submit
Clear
Done
Internet


Database Query - Microsoft Internet Explorer
File Edit
Address http://127.0.0.1/
Go
Links
Data Base Query
Thank you Mr. Ali Abdallah
Here is the information you requested
Maker | Model Type | Model Year | Color | Price
Ford | Tempo | 95 | paige | 50000
Ford | Mercury | 96 | white | 40000
Toyota | Sunny | 97 | brown | 32000
Toyota | Cressida | 96 | blue | 30000
Nissan | Cedric | 95 | green | 30000
Nissan | Ultima | 96 | grey | 30000
Nissan | Ultima | 94 | silver | 25000
Done
Internet