Dr. Deepak Mathur

ADO.NET

DATABASE

- Importance of Data
- Definition of Data Base
- Need of DataBase
- Need of DataBase Technology

- First data access model, DAO (data access model)
- Created for local databases with the built-in Jet engine
- Had performance and functionality issues
- Next came RDO (Remote Data Object) and ADO (Active Data Object)
- Both were designed for Client Server architectures

- Soon ADO took over RDO.
- ADO was a good architecture

But ADO had some problems such as

1.With ADO, all the data is contained in a recordset object which had problems when implemented on

1. Network
2. Penetrating firewalls

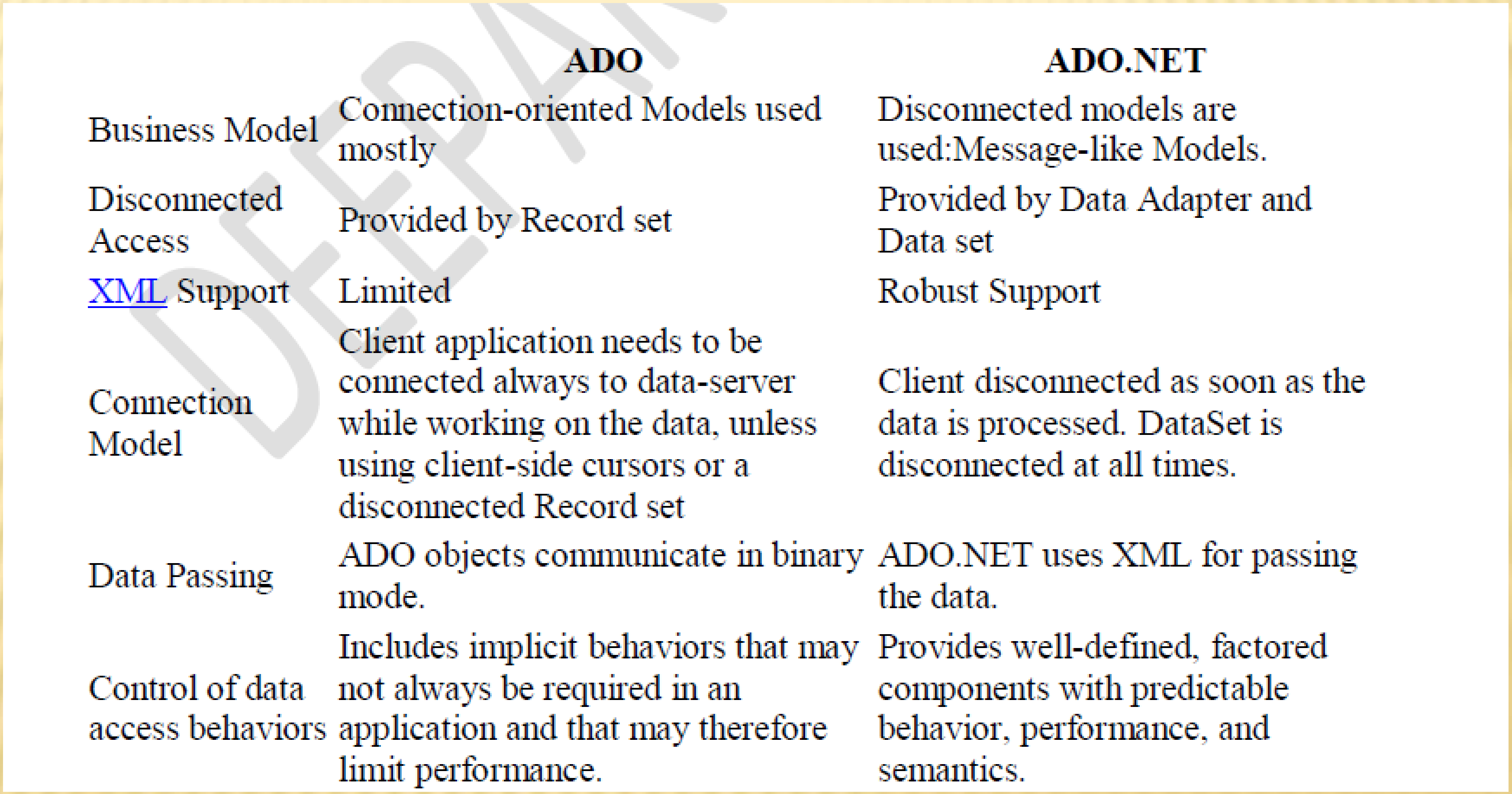
Another problem

2. ADO was a connected data access, which means that when a connection to the database is established the connection remains open until the application is closed. It raises

1.Database security issue

2. Network traffic issue whose example is 10 persons open connection vs 100 persons open conn

- Problems of ADO were somewhat reduced in ADO.NET
- Before discussing ADO.NET let us discuss comparison between ADO and ADO.NET



	ADO	**ADO.NET**
Business Model	Connection-oriented Models used mostly	Disconnected models are used:Message-like Models.
Disconnected Access	Provided by Record set	Provided by Data Adapter and Data set
XML Support	Limited	Robust Support
Connection Model	Client application needs to be connected always to data-server while working on the data, unless using client-side cursors or a disconnected Record set	Client disconnected as soon as the data is processed. DataSet is disconnected at all times.
Data Passing	ADO objects communicate in binary mode.	ADO.NET uses XML for passing the data.
Control of data access behaviors	Includes implicit behaviors that may not always be required in an application and that may therefore limit performance.	Provides well-defined, factored components with predictable behavior, performance, and semantics.

THE ADO.NET DATA ARCHITECTURE

- Data Access in ADO.NET relies on two components:
- DataSet
- Data Provider.

DATASET

- The dataset is a disconnected, in-memory representation of data.
- It can be considered as a local copy of the relevant portions of the database.
- The DataSet is persisted in memory and the data in it can be **manipulated and updated**
- And changes **can be made back to the central database** for updating.

- The data in DataSet can be loaded from any valid data source like
  - Microsoft SQL server database
  - Oracle database
  - Microsoft Access database.

DATA PROVIDER

- DataProvider is a set of related components that work together to provide data in an efficient and performance driven manner
- The Data Provider is responsible for providing and maintaining the connection to the database.

DATA PROVIDER TYPES

- The .NET Framework currently comes with two DataProviders:
- SQL Data Provider which is designed only to work with Microsoft's SQL Server 7.0 or later and the
- OleDb DataProvider which allows us to connect to other types of databases like Access and Oracle.

- Each DataProvider consists of the following component classes:
- The Connection object which provides a connection to the database
- The Command object which is used to execute a command .
- The DataReader object which provides a forward-only, read only, connected recordset .
- The DataAdapter object which populates a disconnected DataSet with data and performs update

THE CONNECTION OBJECT

- The Connection object creates the connection to the database.
- The Connection object contains all of the information required to open a connection to the database.
- Microsoft Visual Studio .NET provides two types of Connection classes:
  - SqlConnection object, which is designed specifically to connect to Microsoft SQL Server 7.0 or later
  - OleDbConnection object, which can provide connections to a wide range of database types like Microsoft Access and Oracle.

```
Dim connetionString As String
Dim cnn As OleDbConnection

 connetionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Your mdb filename;"
 cnn = New OleDbConnection(connetionString)
 cnn.Open()
```

THE COMMAND OBJECT

- Command objects are used to execute commands to a database
- The Command object is represented by two corresponding classes: SqlCommand and OleDbCommand
- The Command objects can be used to execute
  - Stored procedures
  - SQL commands
  - Return complete tables

COMMAND OBJECT METHODS

- Command objects provide three methods that are used to execute commands on the database and these are:
  - ExecuteNonQuery: Executes commands that have no return values such as INSERT, UPDATE or DELETE
  - ExecuteScalar: Returns a single value from a database query
  - ExecuteReader: Returns a result set by way of a DataReader object

DATAREADER OBJECT

- The DataReader object provides a forward-only, read-only, connected stream recordset from a database.
- DataReader objects cannot be directly instantiated.
- Rather, the DataReader is returned as the result of the Command object's ExecuteReader method.

THE DATAREADER OBJECT

- When u will fire ExecuteReader method then that selectquery data would be stored in data reader

  Again the data reader would be read only

  The SqlCommand.ExecuteReader method returns a SqlDataReader object, and the OleDbCommand.ExecuteReader method returns an OleDbDataReader object.

- What if further manipulation of data is required.
- Where changes can be made and written back to the database.
- Answer is in next slide

DATAADAPTER OBJECT

- It is essentially the middleman facilitating all communication between the database and a DataSet.
- The DataAdapter is used either to fill a DataTable or DataSet with data from the database with it's Fill method.
- After the memory-resident data has been manipulated, the DataAdapter can commit the changes to the database by calling the Update method.

- The DataAdapter provides four properties that represent database commands:

SelectCommand
InsertCommand
DeleteCommand
UpdateCommand

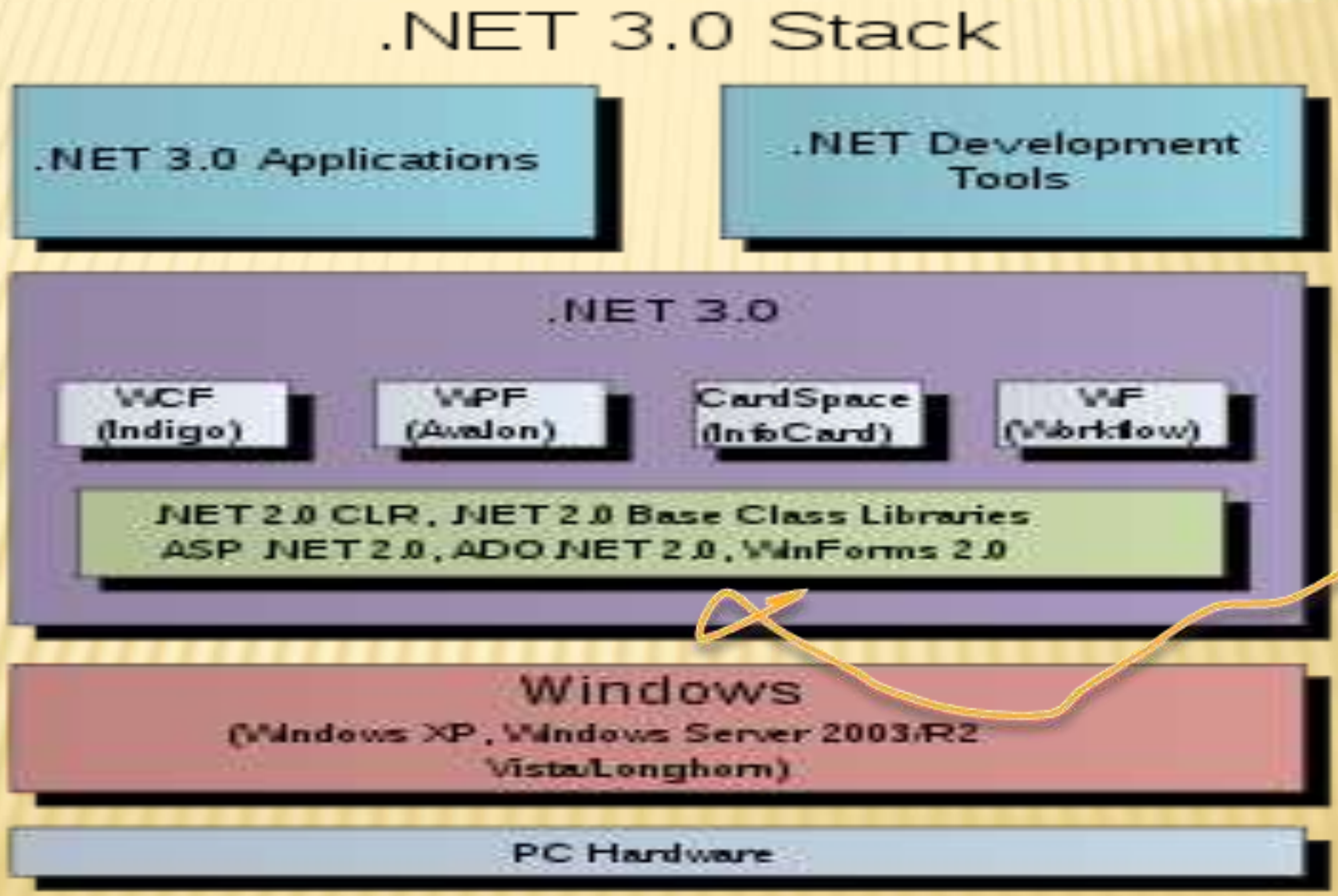
- That is

changes in the DataSet are copied back to the database and the appropriate InsertCommand, DeleteCommand, or UpdateCommand is executed.

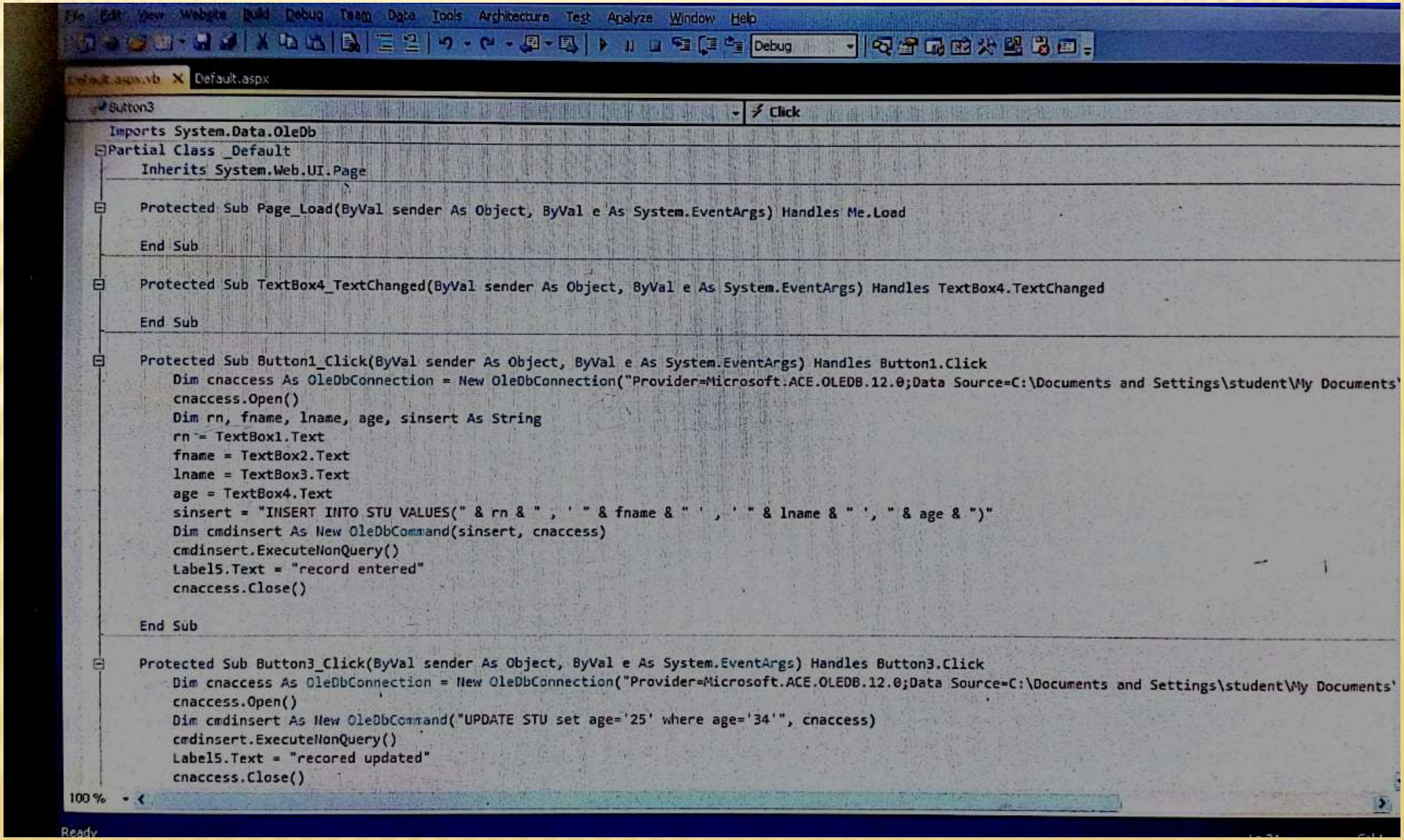
SUMMARY

- A connection object establishes the connection for the application with the database.
- The command object provides direct execution of the command to the database.
  - If the command returns more than a single value, the command object returns a DataReader to provide the data.
  - Alternatively, the DataAdapter can be used to fill the Dataset object. The database can be updated using the command object or the DataAdapter.

.NET ARCHITECTURE SHOWING ADO.NET

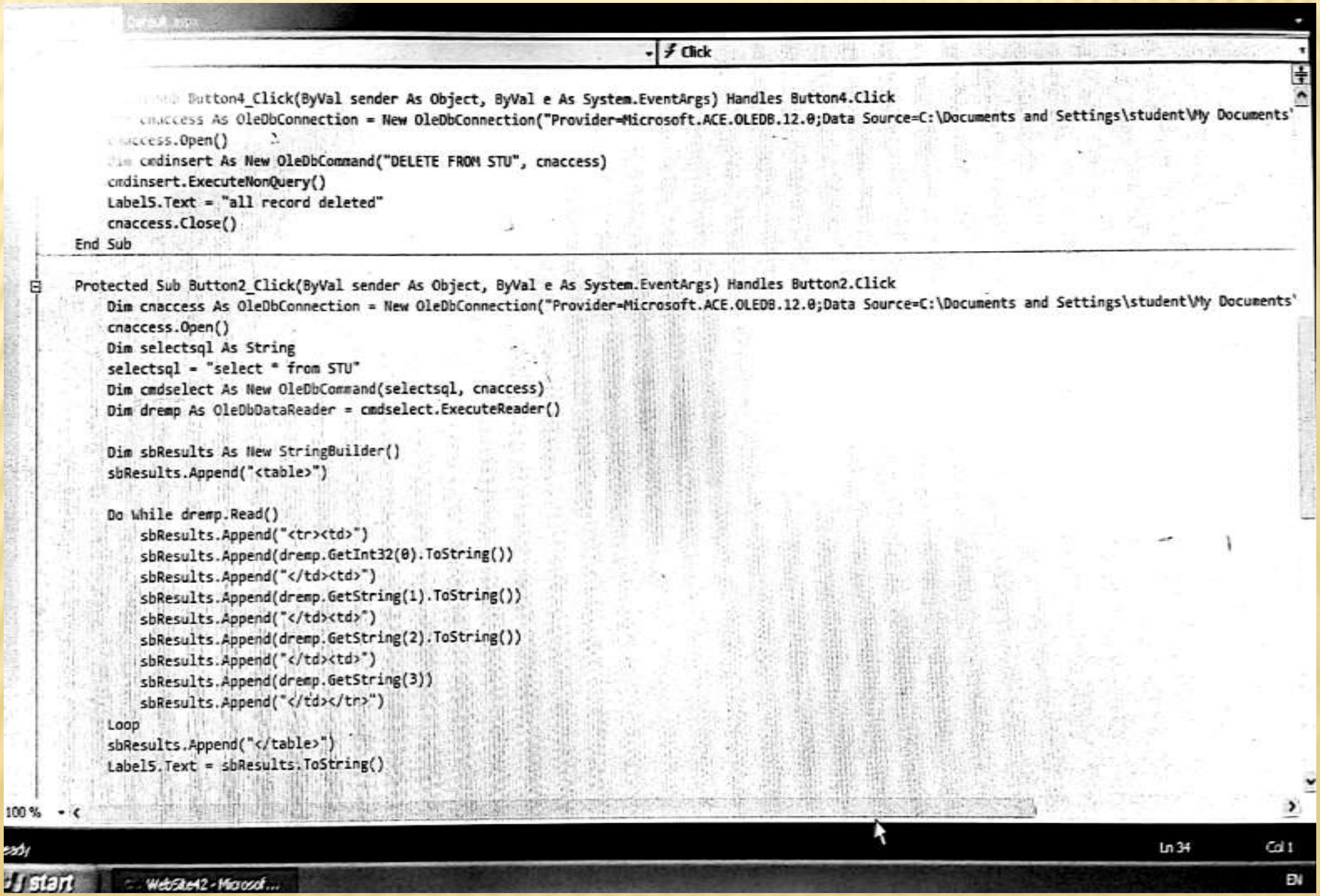


DATACOMMAND


File Edit View Website Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Default.aspx.vb
Default.aspx
Button3
Click
100 %
Ready

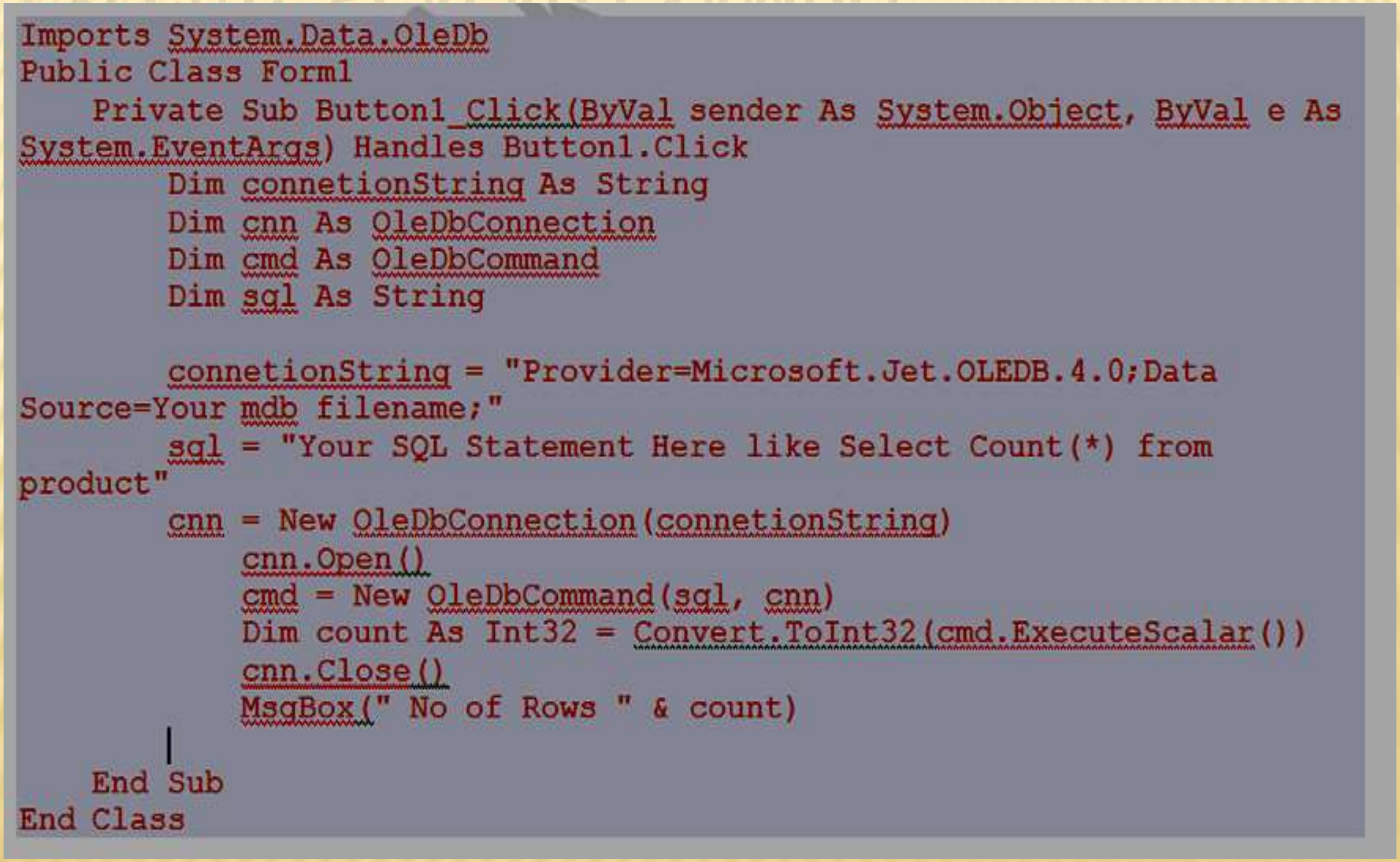
```
Imports System.Data.OleDb
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

    End Sub

    Protected Sub TextBox4_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles TextBox4.TextChanged

    End Sub

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim cnaccess As OleDbConnection = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\student\My Documents'
        cnaccess.Open()
        Dim rn, fname, lname, age, sinsert As String
        rn = TextBox1.Text
        fname = TextBox2.Text
        lname = TextBox3.Text
        age = TextBox4.Text
        sinsert = "INSERT INTO STU VALUES(" & rn & " , ' " & fname & " ' , ' " & lname & " ', " & age & ")"
        Dim cmdinsert As New OleDbCommand(sinsert, cnaccess)
        cmdinsert.ExecuteNonQuery()
        Label5.Text = "record entered"
        cnaccess.Close()

    End Sub

    Protected Sub Button3_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button3.Click
        Dim cnaccess As OleDbConnection = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\student\My Documents'
        cnaccess.Open()
        Dim cmdinsert As New OleDbCommand("UPDATE STU set age='25' where age='34'", cnaccess)
        cmdinsert.ExecuteNonQuery()
        Label5.Text = "recored updated"
        cnaccess.Close()
```

DATACOMMAND CONTD



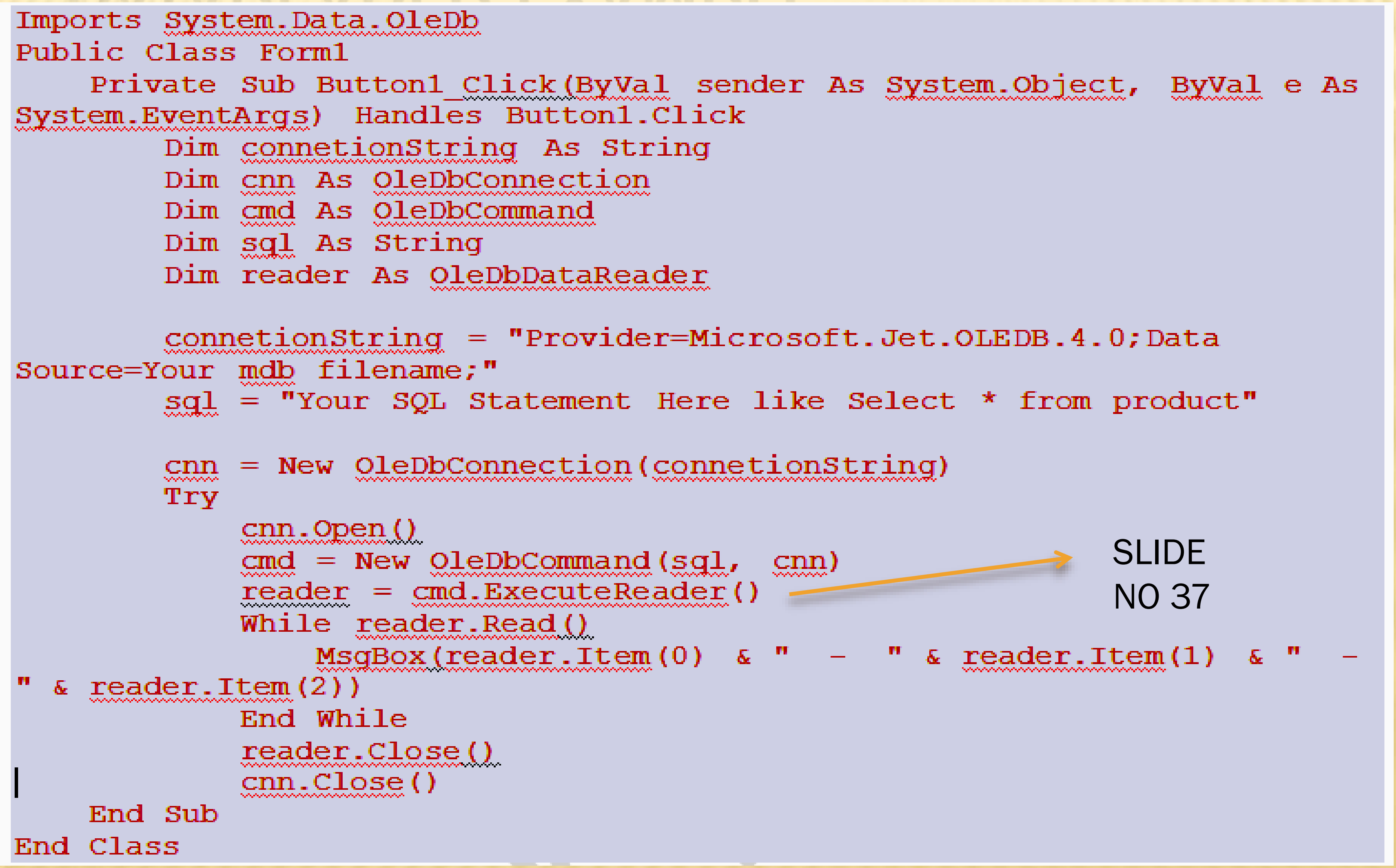
```
    Protected Sub Button4_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button4.Click
        Dim cnaccess As OleDbConnection = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\student\My Documents'
        cnaccess.Open()
        Dim cmdinsert As New OleDbCommand("DELETE FROM STU", cnaccess)
        cmdinsert.ExecuteNonQuery()
        Label5.Text = "all record deleted"
        cnaccess.Close()
    End Sub

    Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click
        Dim cnaccess As OleDbConnection = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\student\My Documents'
        cnaccess.Open()
        Dim selectsql As String
        selectsql = "select * from STU"
        Dim cmdselect As New OleDbCommand(selectsql, cnaccess)
        Dim dremp As OleDbDataReader = cmdselect.ExecuteReader()

        Dim sbResults As New StringBuilder()
        sbResults.Append("<table>")

        Do While dremp.Read()
            sbResults.Append("<tr><td>")
            sbResults.Append(dremp.GetInt32(0).ToString())
            sbResults.Append("</td><td>")
            sbResults.Append(dremp.GetString(1).ToString())
            sbResults.Append("</td><td>")
            sbResults.Append(dremp.GetString(2).ToString())
            sbResults.Append("</td><td>")
            sbResults.Append(dremp.GetString(3))
            sbResults.Append("</td></tr>")
        Loop
        sbResults.Append("</table>")
        Label5.Text = sbResults.ToString()
```

EXECUTE SCALAR EXAMPLE



```
Imports System.Data.OleDb
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim connetionString As String
        Dim cnn As OleDbConnection
        Dim cmd As OleDbCommand
        Dim sql As String

        connetionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Your mdb filename;"
        sql = "Your SQL Statement Here like Select Count(*) from
product"
        cnn = New OleDbConnection(connetionString)
            cnn.Open()
            cmd = New OleDbCommand(sql, cnn)
            Dim count As Int32 = Convert.ToInt32(cmd.ExecuteScalar())
            cnn.Close()
            MsgBox(" No of Rows " & count)

    End Sub
End Class
```

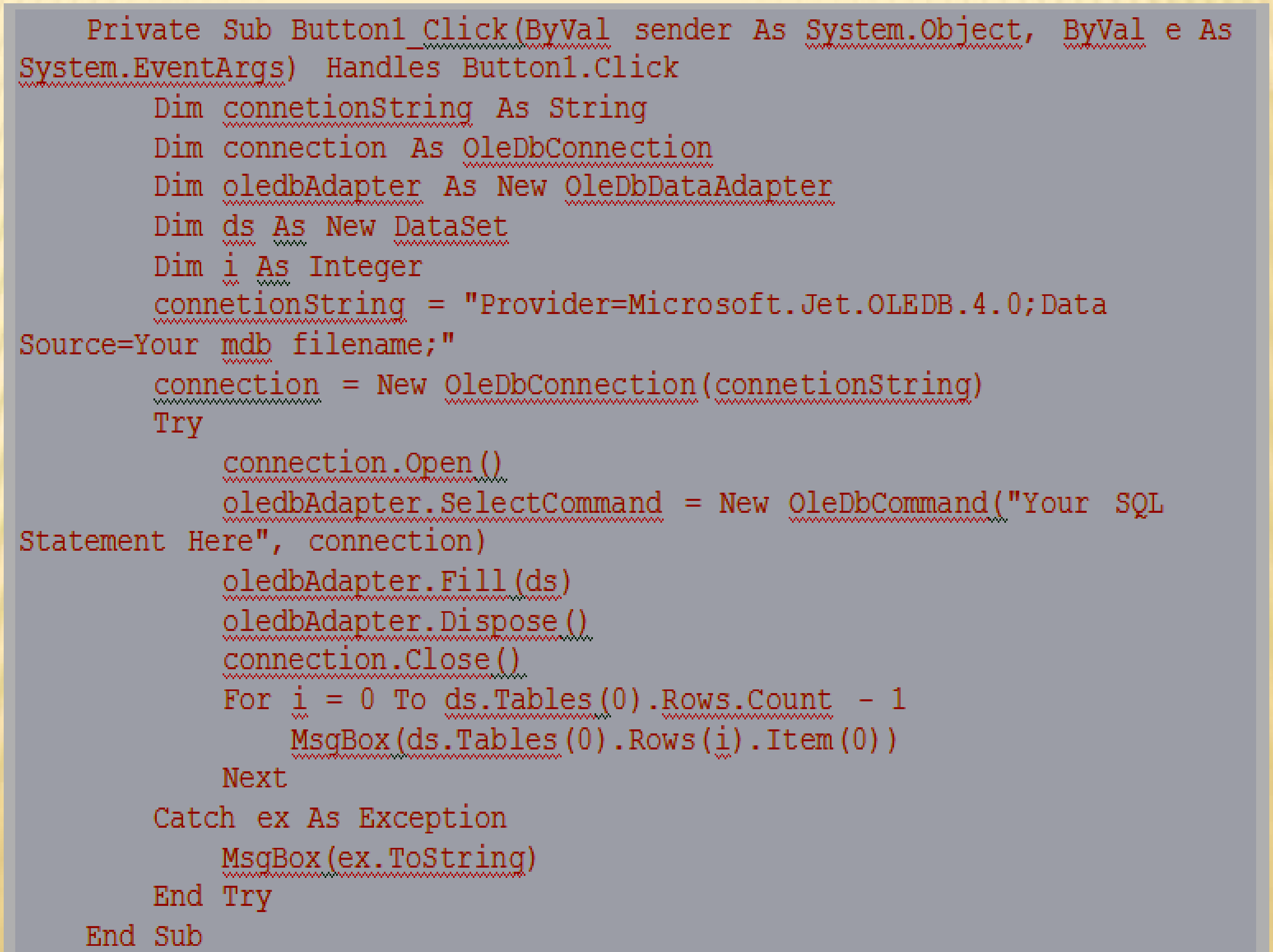
DATAREADER EXAMPLE



```
Imports System.Data.OleDb
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim connetionString As String
        Dim cnn As OleDbConnection
        Dim cmd As OleDbCommand
        Dim sql As String
        Dim reader As OleDbDataReader

        connetionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Your mdb filename;"
        sql = "Your SQL Statement Here like Select * from product"

        cnn = New OleDbConnection(connetionString)
        Try
            cnn.Open()
            cmd = New OleDbCommand(sql, cnn)
            reader = cmd.ExecuteReader()
            While reader.Read()
                MsgBox(reader.Item(0) & "  -  " & reader.Item(1) & "  -
" & reader.Item(2))
            End While
            reader.Close()
            cnn.Close()
    End Sub
End Class
```

SLIDE NO 37

DATA ADAPTER EXAMPLE



```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim connetionString As String
        Dim connection As OleDbConnection
        Dim oledbAdapter As New OleDbDataAdapter
        Dim ds As New DataSet
        Dim i As Integer
        connetionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Your mdb filename;"
        connection = New OleDbConnection(connetionString)
        Try
            connection.Open()
            oledbAdapter.SelectCommand = New OleDbCommand("Your SQL
Statement Here", connection)
            oledbAdapter.Fill(ds)
            oledbAdapter.Dispose()
            connection.Close()
            For i = 0 To ds.Tables(0).Rows.Count - 1
                MsgBox(ds.Tables(0).Rows(i).Item(0))
            Next
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub
```

DATA BINDING

- ASP.NET provides a rich set of controls that enable you to display information to users.
- *Data binding means binding controls to information stored in a data store.*
- A data store can be as simple as a public property on a page, or as complex as a database stored on a server

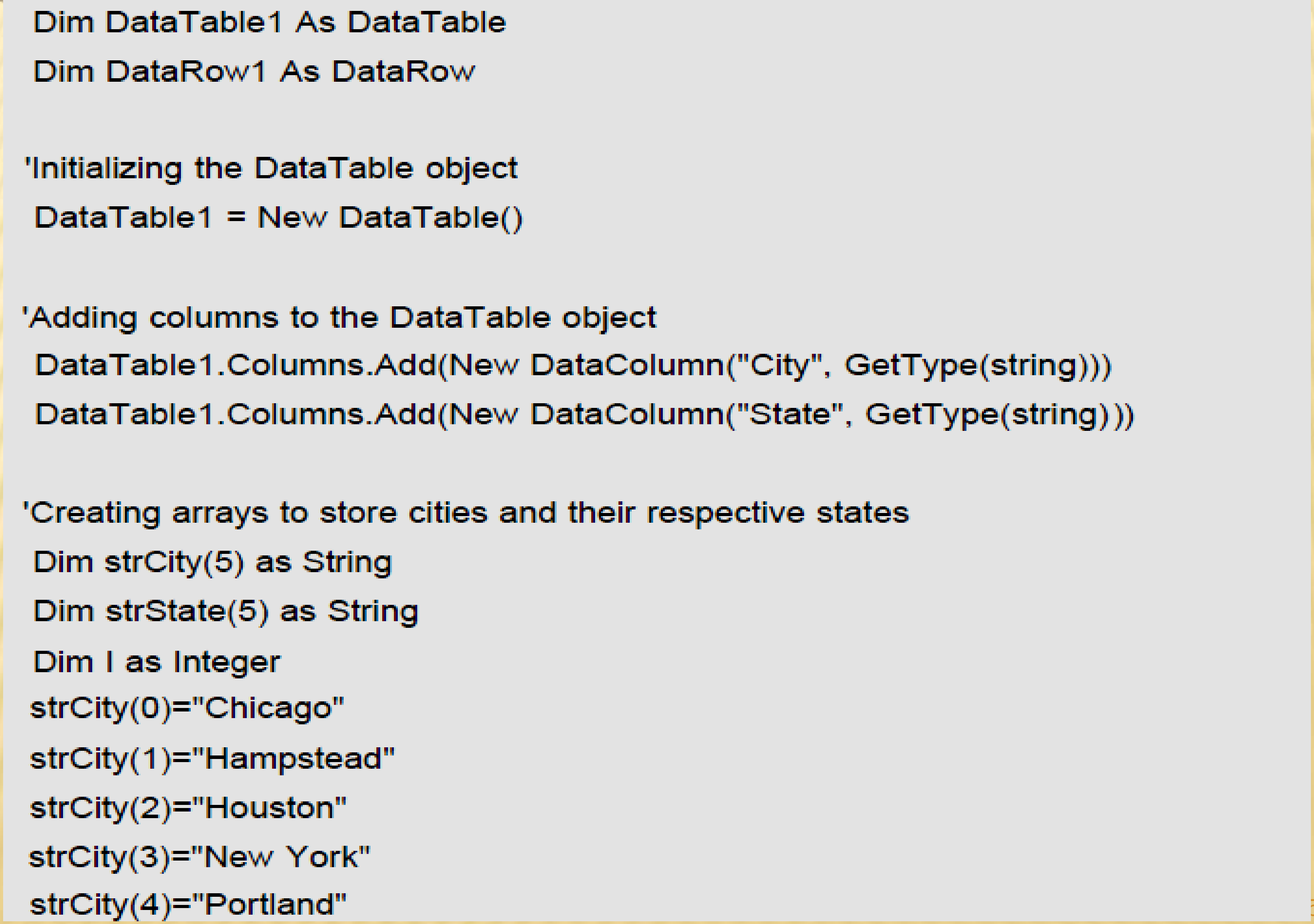
- Data binding gives extensive control over data.
- It refers to the process of assigning a value to a property of a runtime. For example, you can use data binding to bind the properties of a control to a data source such as the contents of a SQL Server database OR OLEDB database table.
- Any type of data can be bounded to any control or property of control on an ASP.NET page.

BINDING TO A DATAVIEW

- The DataView class represents a custom view of a data table.
- This class is a member of the System.Data namespace, and to use this class in your page.
- In this form, add a DataGrid control.

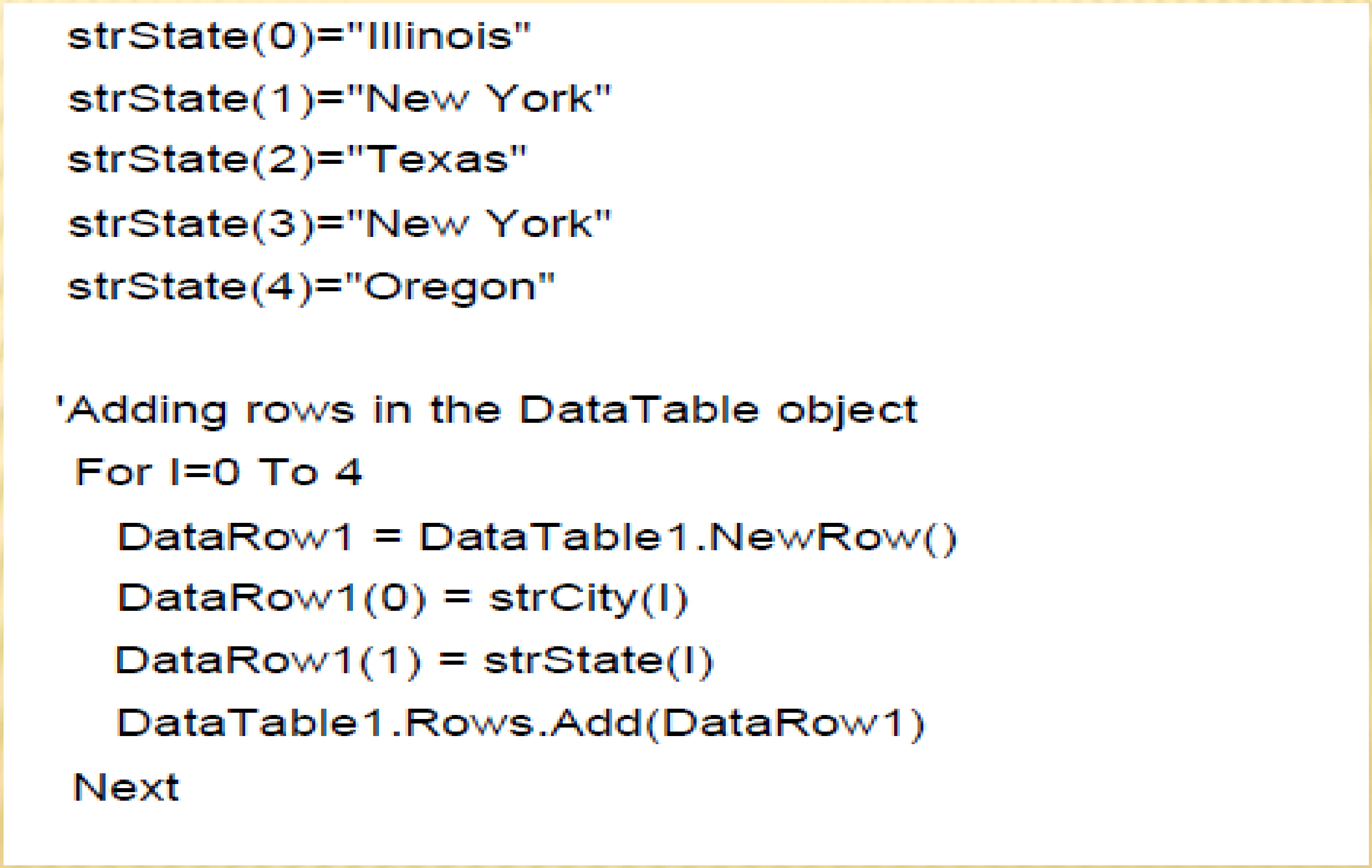
- You can bind a DataView object to a DataGrid control.
- A DataGrid control displays information in row and column format. In this section, you'll create an object of the DataView class.
- This object represents a data table that displays cities and their respective states.
- Then, you'll bind this DataView object to the DataSource property of the DataGrid control.

CODE



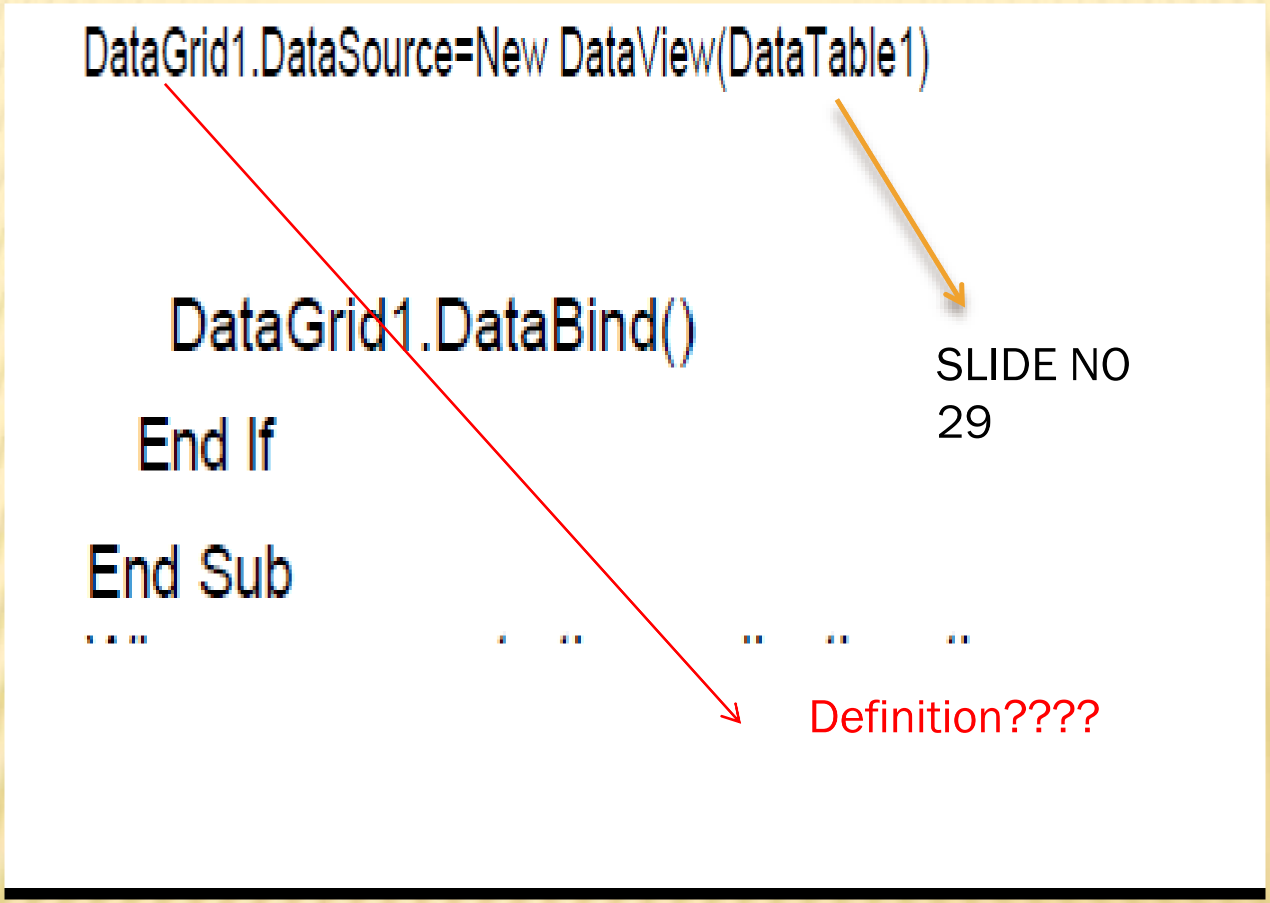
```
 Dim DataTable1 As DataTable
 Dim DataRow1 As DataRow

'Initializing the DataTable object
 DataTable1 = New DataTable()

'Adding columns to the DataTable object
 DataTable1.Columns.Add(New DataColumn("City", GetType(string)))
 DataTable1.Columns.Add(New DataColumn("State", GetType(string)))

'Creating arrays to store cities and their respective states
 Dim strCity(5) as String
 Dim strState(5) as String
 Dim I as Integer
strCity(0)="Chicago"
strCity(1)="Hampstead"
strCity(2)="Houston"
strCity(3)="New York"
strCity(4)="Portland"
```

CODE CONTD..



```
 strState(0)="Illinois"
 strState(1)="New York"
 strState(2)="Texas"
 strState(3)="New York"
 strState(4)="Oregon"

'Adding rows in the DataTable object
 For I=0 To 4
    DataRow1 = DataTable1.NewRow()
    DataRow1(0) = strCity(I)
    DataRow1(1) = strState(I)
    DataTable1.Rows.Add(DataRow1)
 Next
```

CODE CONTD………..

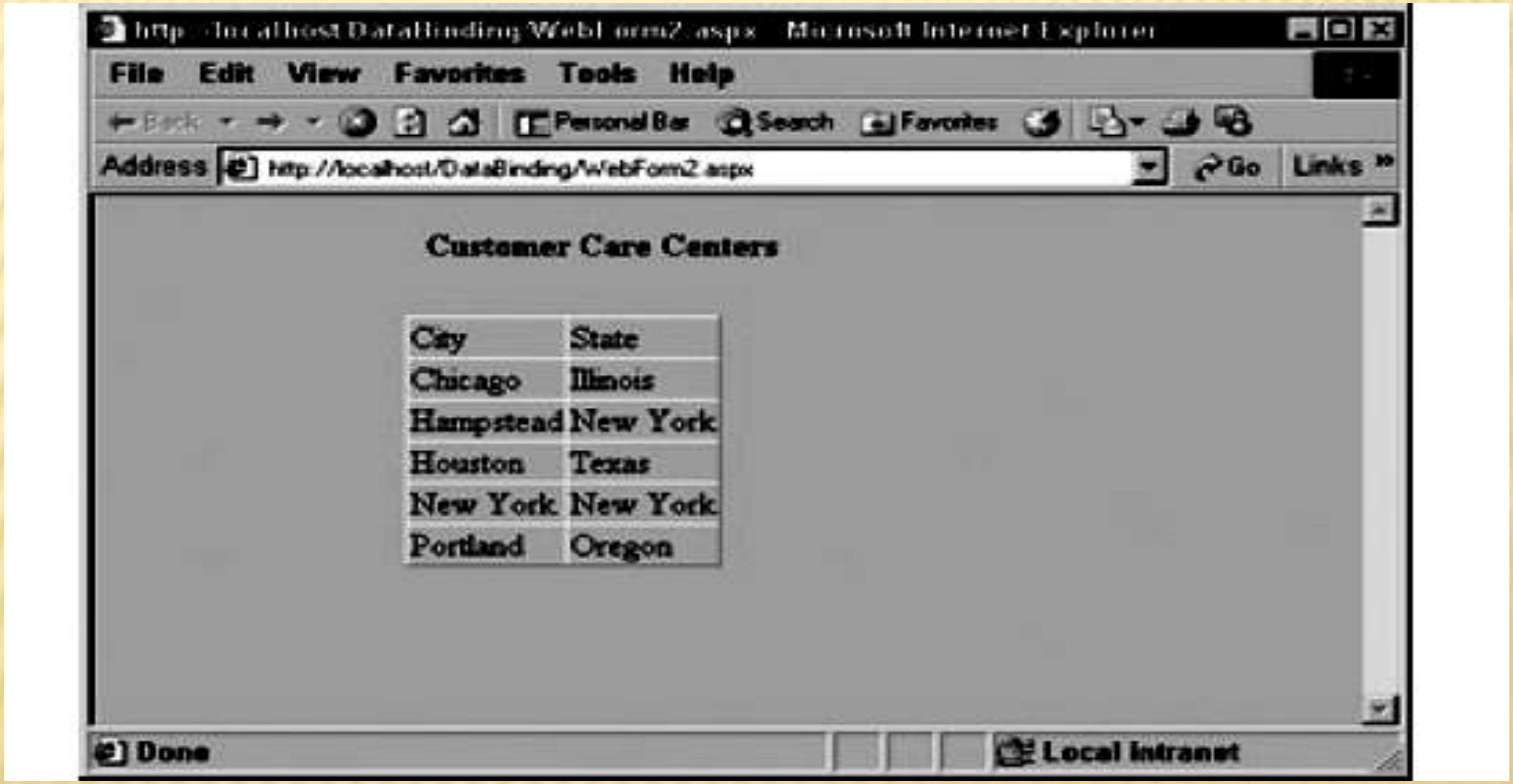


```
DataGrid1.DataSource=New DataView(DataTable1)


    DataGrid1.DataBind()

  End If

End Sub
```

SLIDE NO 29

Definition????

OUTPUT


http://localhost/DataBinding/WebForm2.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Bar Search Favorites
Address http://localhost/DataBinding/WebForm2.aspx
Go Links
Done
Local intranet

Customer Care Centers

City	State
Chicago	Illinois
Hampstead	New York
Houston	Texas
New York	New York
Portland	Oregon

DATA BINDING

- ASP.NET provides a rich set of controls that enable you to display information to users.
- *Data binding means binding controls to information stored in a data store.*
- A data store can be as simple as a public property on a page, or as complex as a database stored on a server

- You can bind a control to different data stores, such as properties, methods, or collections.
- Simple controls are the controls that can bind only to a single value. Some simple controls include Label, TextBox, and Buttoncontrols.

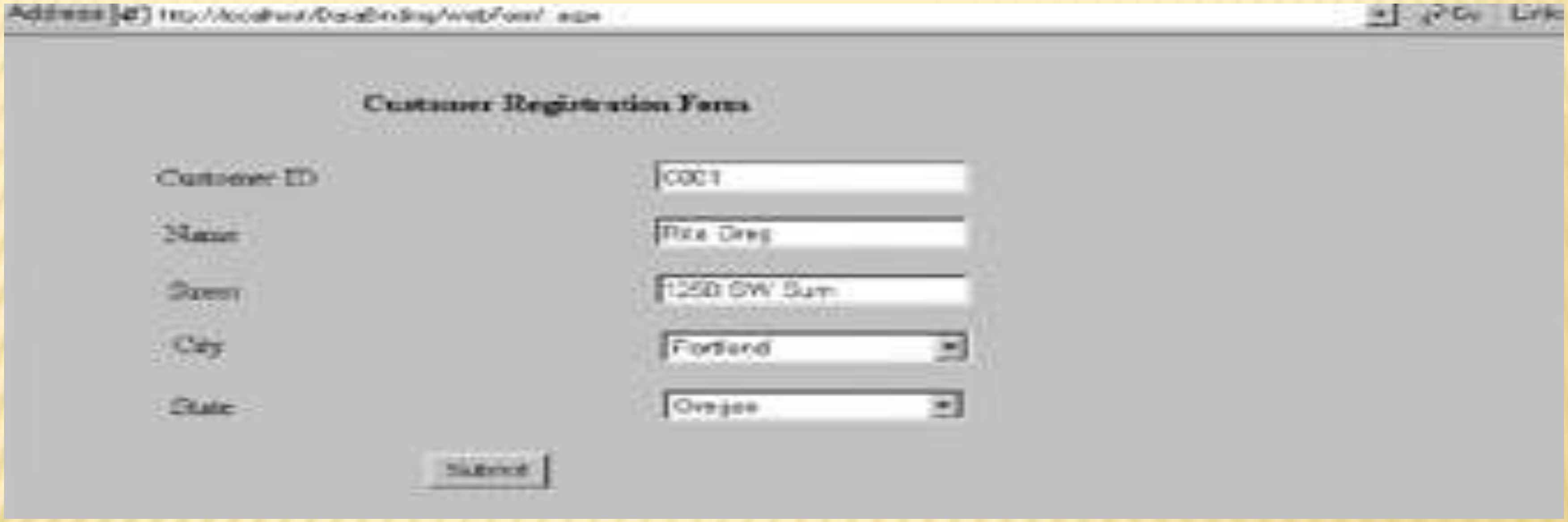
DATA BINDING TO AN ARRAYLIST

- this section, you'll bind the ArrayList class to the DataSource property of the DropDownList with ID CustState

- Dim objArrayList as ArrayList= new ArrayList()
- objArrayList.Add ("New York")
- objArrayList.Add ("California")
- objArrayList.Add ("Oregon")
- objArrayList.Add ("Illinois")
- objArrayList.Add ("Texas")
- objArrayList.Add ("None")

- CustState.DataSource = objArrayList
- CustState.DataBind()

OUTPUT



Address http://localhost/DataBinding/WebForm1.aspx

Customer Registration Form

Customer ID	C001
Name	Rita Greg
Street	1250 SW Sum
City	Portland
State	Oregon

Now what if this is written ??????

CustState.selectedItem.Text

- THANK U

TAB STRIP , MASTER PAGE AND NAVIGATION CONTROL

TABSTRIP AND MULTIPAGE CONTROLS

- The TabStrip control is used to present tabbed controls, which can be used along with the MultiPage control to display varied information in a given space
- Users can click to switch between the different tabs
- The MultiPage control is used to display multiple pages of data in a given screen area.

TABSTRIP CONTROL

```
<tagprefix:TabStrip runat="Server"
   TabDefaultStyle=".." TabHoverStyle=
".." TabSelectedStyle=".." SepDefaultStyle="..">
<tagprefix:Tab text=".." >
<tagprefix:Tab text="Node1.1"/>
<tagprefix:Tab text="Node1.2">
</tagprefix:Tab>
</tagprefix:TabStrip>
```

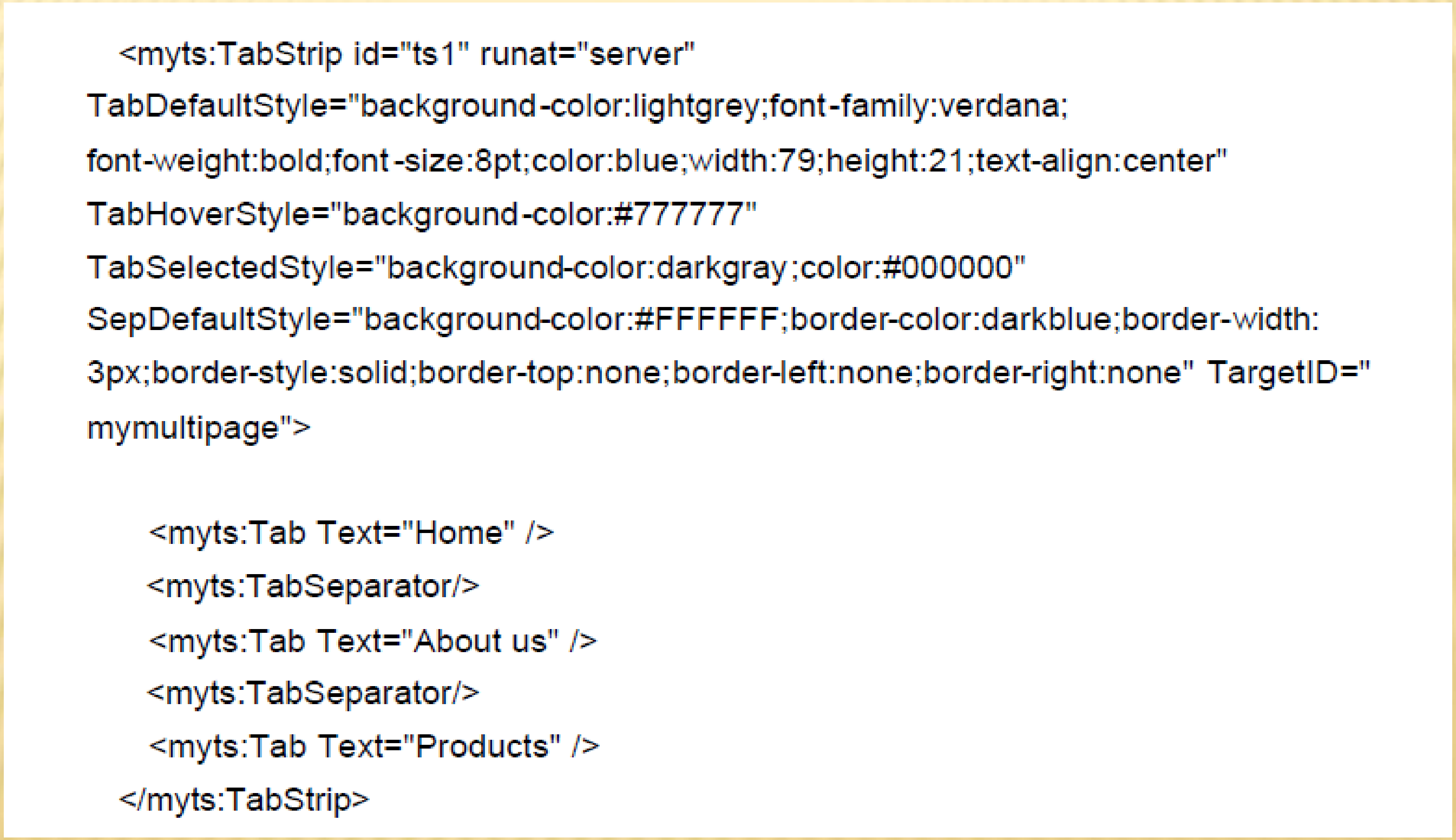
- TabStrip: Defines a TabStrip control, which acts as a container for the tabs and tab separators.
- Tab: Defines a tab element in the TabStrip control, which is rendered on the client browser as tabs on top of the tab strip

MULTIPAGE CONTROL

- The MultiPage control is a container control that contains a set of PageView elements
- The MultiPage control is typically used with the TabStrip control to give users the ability to navigate from one page to another

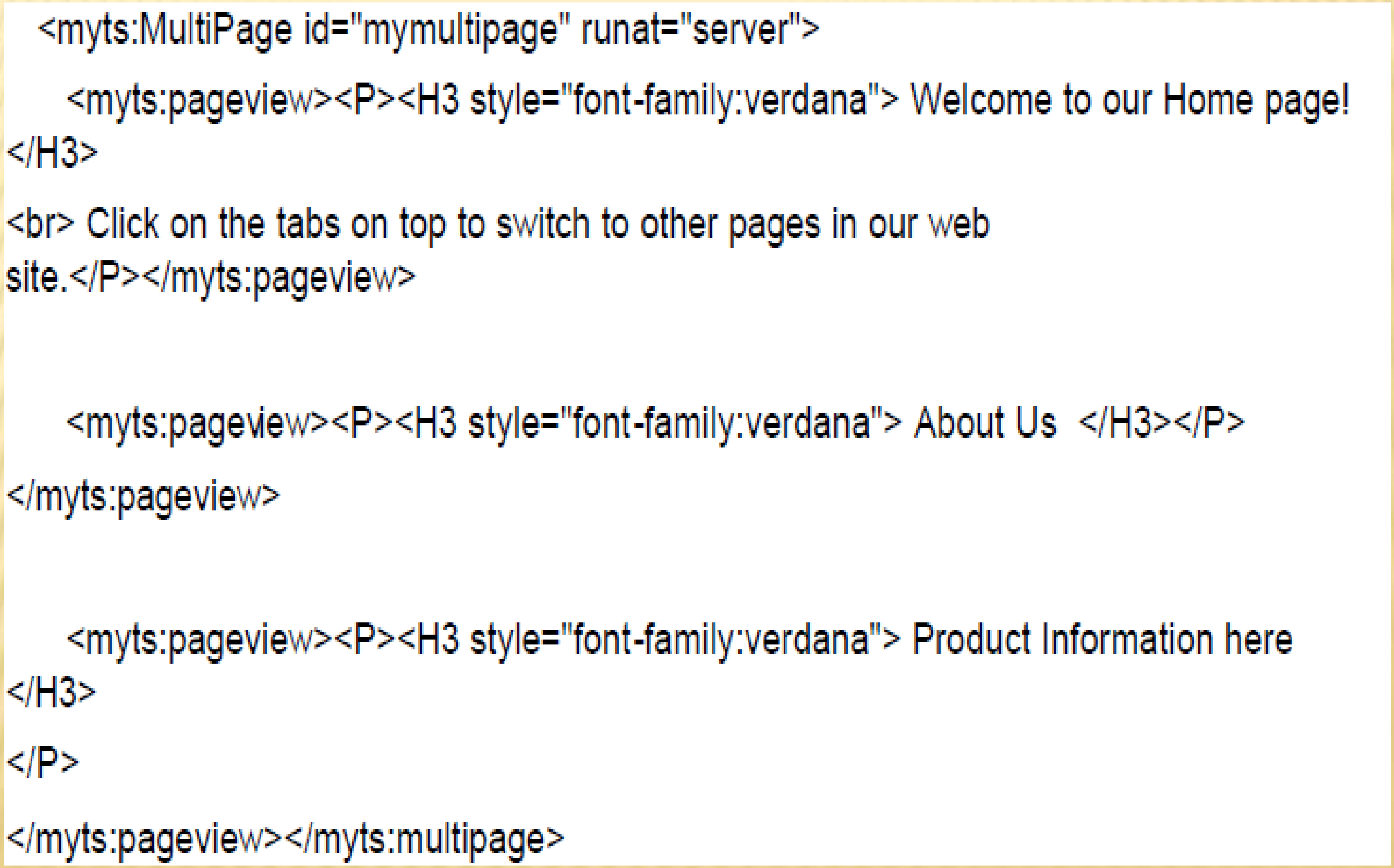
- <tagprefix:MultiPage runat="server" selectedindex="1">
- <tagprefix:PageView>
- <P> Data for page view <B>1</B> </P>
- </tagprefix:PageView>
- <tagprefix:PageView>
- <P> Data for page view <B>2</B> </P>
- </tagprefix:PageView>
- </tagprefix:MultiPage>

PRACTICAL CODE <MAKING 3 TABS>



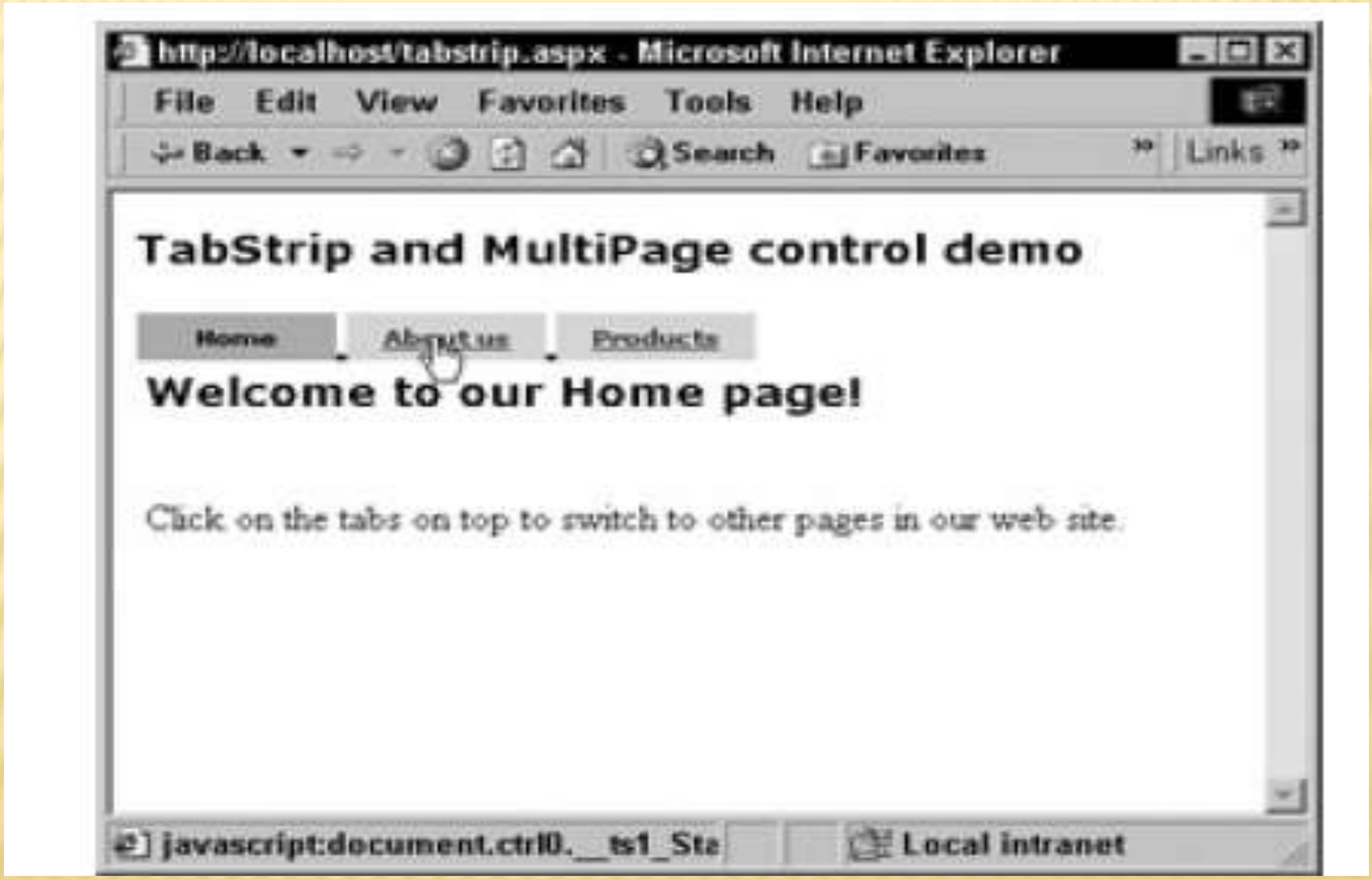
```
   <myts:TabStrip id="ts1" runat="server"
TabDefaultStyle="background-color:lightgrey;font-family:verdana;
font-weight:bold;font-size:8pt;color:blue;width:79;height:21;text-align:center"
TabHoverStyle="background-color:#777777"
TabSelectedStyle="background-color:darkgray;color:#000000"
SepDefaultStyle="background-color:#FFFFFF;border-color:darkblue;border-width:
3px;border-style:solid;border-top:none;border-left:none;border-right:none" TargetID="
mymultipage">

      <myts:Tab Text="Home" />
      <myts:TabSeparator/>
      <myts:Tab Text="About us" />
      <myts:TabSeparator/>
      <myts:Tab Text="Products" />
   </myts:TabStrip>
```

PRACTICAL CODE <MAKING 3 PAGEVIEWS>



```
  <myts:MultiPage id="mymultipage" runat="server">
    <myts:pageview><P><H3 style="font-family:verdana"> Welcome to our Home page!
</H3>
<br> Click on the tabs on top to switch to other pages in our web
site.</P></myts:pageview>


    <myts:pageview><P><H3 style="font-family:verdana"> About Us  </H3></P>
</myts:pageview>


    <myts:pageview><P><H3 style="font-family:verdana"> Product Information here
</H3>
</P>
</myts:pageview></myts:multipage>
```

OUTPUT


http://localhost/tabstrip.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Links
Home
About us
Products
javascript:document.ctrl0.__ts1_Ste
Local intranet

TabStrip and MultiPage control demo

Welcome to our Home page!

Click on the tabs on top to switch to other pages in our web site.

MASTER PAGES

- Master pages allow you to create a consistent look and behavior for all the pages (or group of pages) in your web application.
- A master page provides a template for other pages, with shared layout and functionality.
- Master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page.

- The content pages contain the content you want to display.
- When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.
-

MASTER PAGE EXAMPLE

```
<%@ Master %>
<html>
<body>
<h1>Standard Header From Masterpage</h1>
<asp:ContentPlaceHolder id="CPH1"
runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

@ Master directive defines it as a master page.

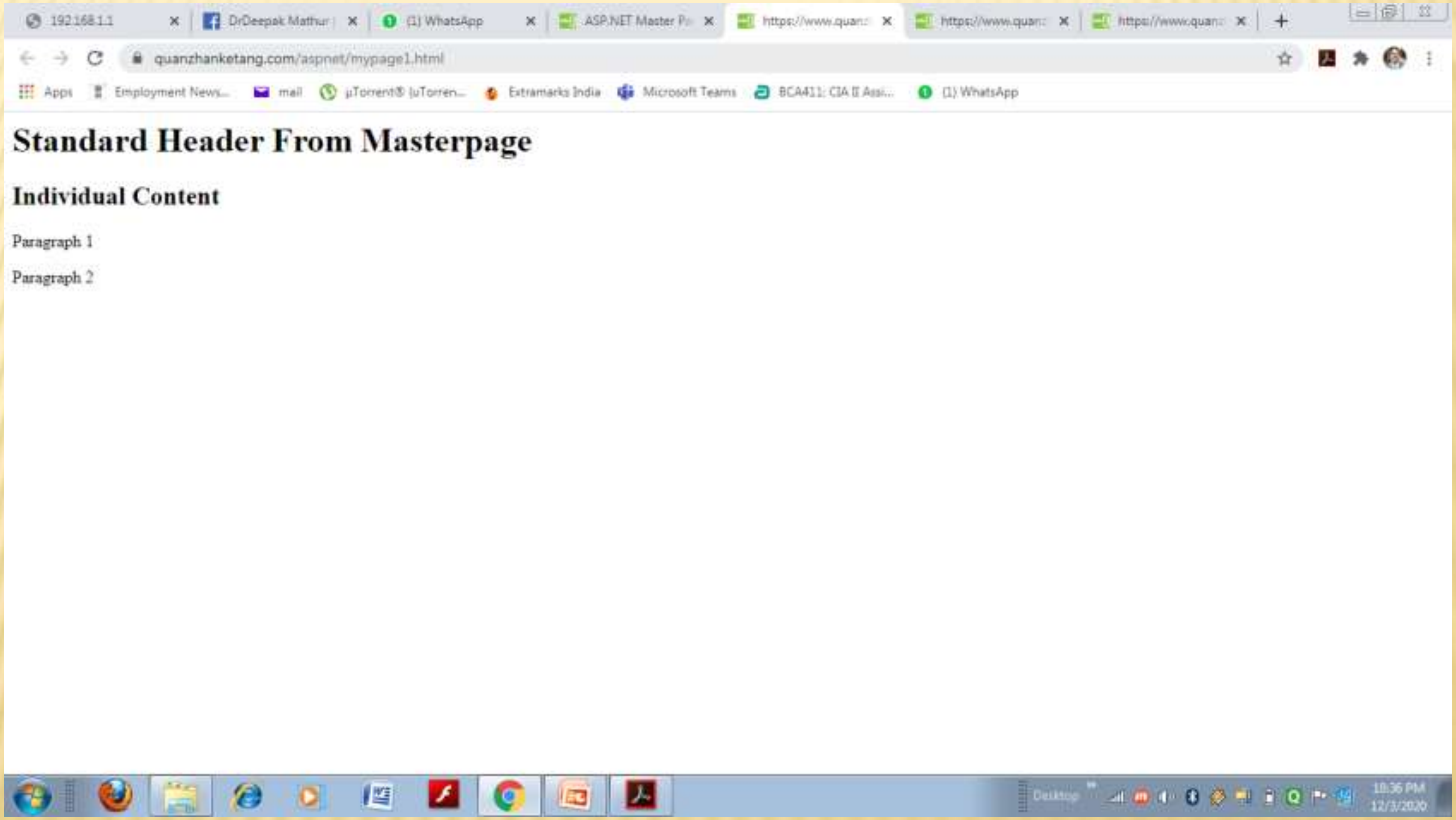
- The master page contains a placeholder tag **<asp:ContentPlaceHolder>** for individual content.
- The **id="CPH1"** attribute identifies the placeholder, allowing many placeholders in the same master page.
- This master page was saved with the name **"master1.master".**

CONTENT PAGE EXAMPLE<"MYPAGE1.ASPX”>

```
<%@ Page MasterPageFile="master1.master"
%>
<asp:Content ContentPlaceHolderId="CPH1"
runat="server">
  <h2>Individual Content</h2>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</asp:Content>
```

- The content page contains a content tag **<asp:Content>** with a reference to the master page (ContentPlaceHolderId="CPH1").

OUTPUT



Standard Header From Masterpage

Individual Content

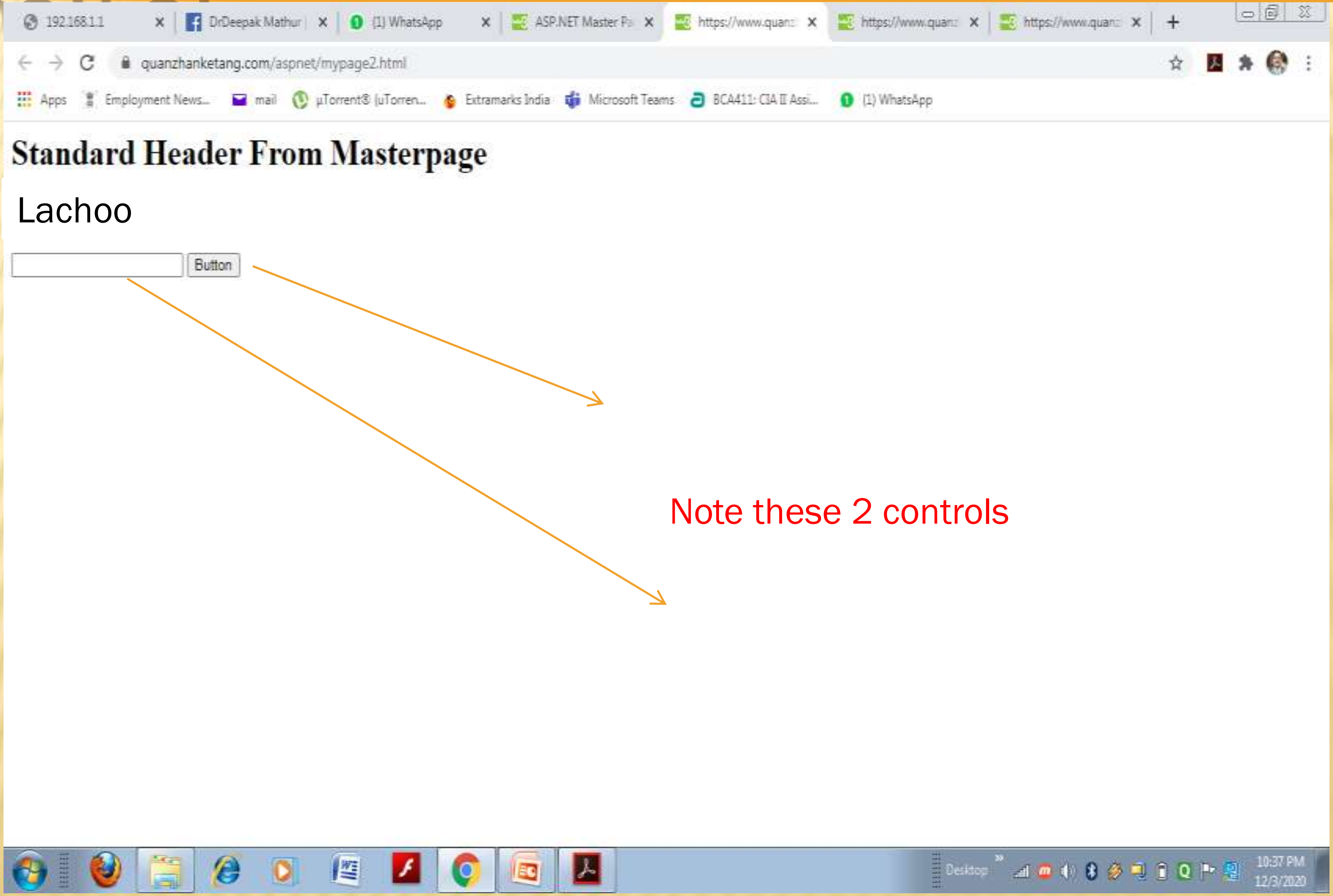
Paragraph 1

Paragraph 2

CONTENT PAGE WITH CONTROLS EXAMPLE

```
<%@ Page MasterPageFile="master1.master" %>
<asp:Content ContentPlaceHolderId="CPH1"
runat="server">
  <h2>Lachoo</h2>
  <form runat="server">
    <asp:TextBox id="textbox1" runat="server" />
    <asp:Button id="button1" runat="server"
text="Button" />
  </form>
</asp:Content>
```

OUTPUT



Standard Header From Masterpage

Lachoo

Note these 2 controls