TOPIC 5 → Database Access in ASP.NET

LEARNING OUTCOME

By the end of this topic, students will be able to:

- ➢ Describe how to Access Database in ASP.Net
- ➢ Retrieve and display data
- ➢ Use ActiveX Data Objects such as DataSet, DataTable, DataRow, DataAdapter, and DataReader objects
- ➢ Write DbCommand and DbConnection Objects
- ➢ Explain File Uploading Procedures
- ➢ Use Ad Rotator
- ➢ Describe Properties and Events of the AdRotator Class
- ➢ Use Calendar Control

Introduction

ASP.NET allows the following sources of data to be accessed and used:

- Databases (e.g., Access, SQL Server, Oracle, MySQL)
- XML documents
- Business Objects
- Flat files

ASP.NET hides the complex processes of data access and provides much higher level of classes and objects through which data is accessed easily. These classes hide all complex coding for connection, data retrieving, data querying, and data manipulation.

ADO.NET is the technology that provides the bridge between various ASP.NET control objects and the backend data source. In this lecture, we will look at data access and working with the data in brief.

Retrieve and display data

It takes two types of data controls to retrieve and display data in ASP.NET:

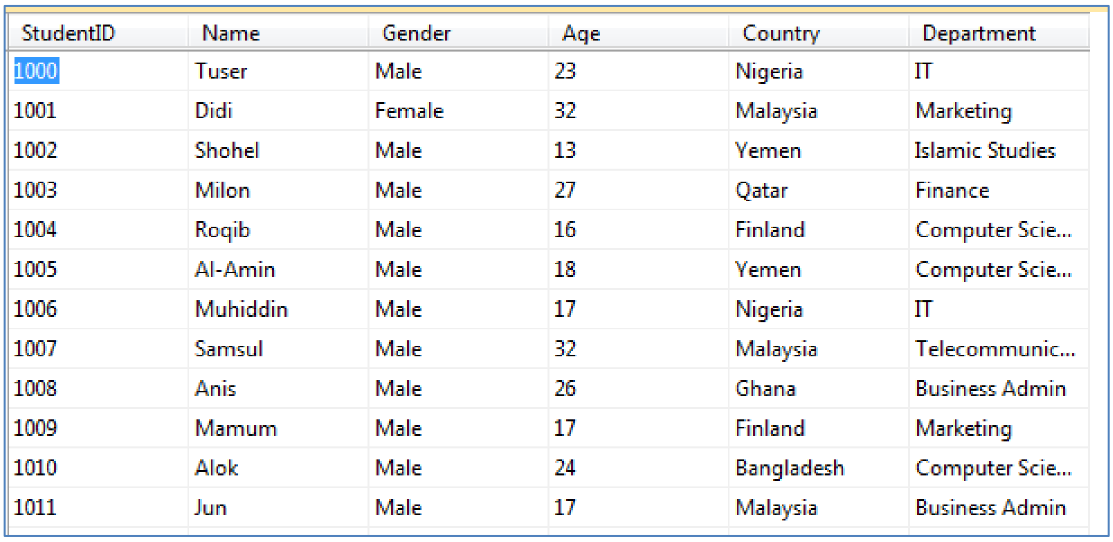
- A data source control - It manages the connection to the data, selection of data, and other jobs such as paging and caching of data etc.
- A data view control - It binds and displays the data and allows data manipulation.

We will discuss the data binding and data source controls in detail later. In this lecture, we will use a SqlDataSource control to access data and a GridView control to display and manipulate data.

We will also use Microsoft SQL Server, which contains the details about LUC students. Name of our database is StudentDB and we will use the data table studentDetails.

The table has the following columns: StudentID, Name, Gender, Age, Country, and Department.

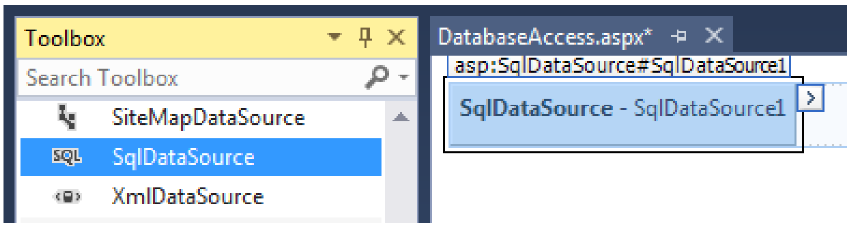
Here is a snapshot of the data table:



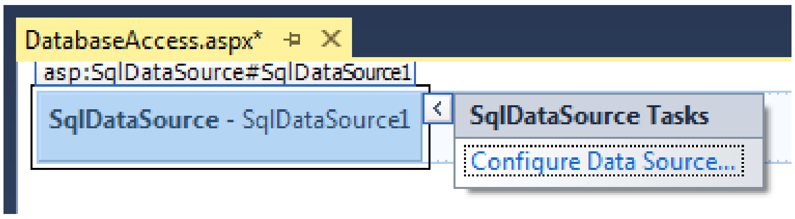
StudentID	Name	Gender	Age	Country	Department
1000	Tuser	Male	23	Nigeria	IT
1001	Didi	Female	32	Malaysia	Marketing
1002	Shohel	Male	13	Yemen	Islamic Studies
1003	Milon	Male	27	Qatar	Finance
1004	Roqib	Male	16	Finland	Computer Scie...
1005	Al-Amin	Male	18	Yemen	Computer Scie...
1006	Muhiddin	Male	17	Nigeria	IT
1007	Samsul	Male	32	Malaysia	Telecommunic...
1008	Anis	Male	26	Ghana	Business Admin
1009	Mamum	Male	17	Finland	Marketing
1010	Alok	Male	24	Bangladesh	Computer Scie...
1011	Jun	Male	17	Malaysia	Business Admin

Let us directly move to action, take the following steps:

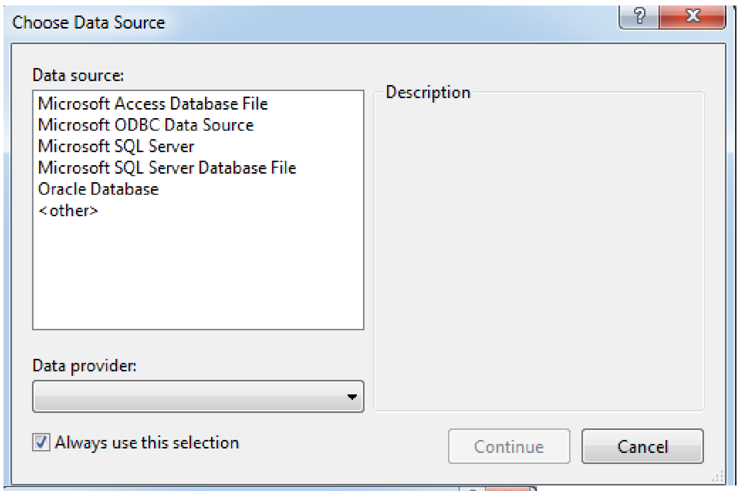
(1) Create a web site and add a SqlDataSourceControl on the web form.



(2) Click on the Configure Data Source option.



(3) Click on the New Connection button to establish connection with a database.


Choose Data Source

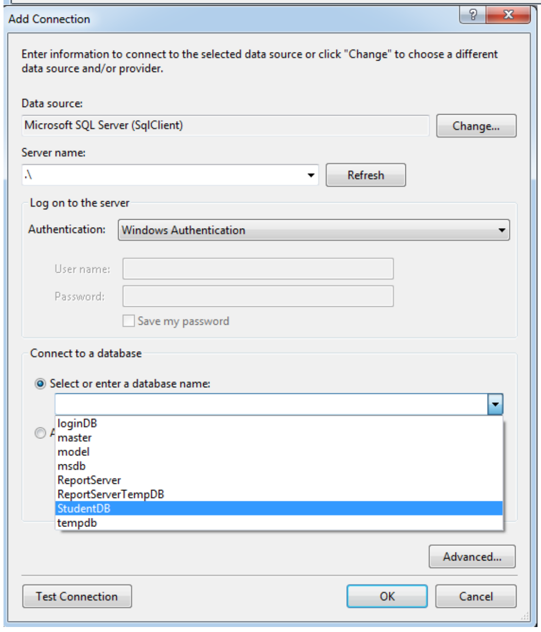
Data source:

- Microsoft Access Database File
- Microsoft ODBC Data Source
- Microsoft SQL Server
- Microsoft SQL Server Database File
- Oracle Database
- <other>

Description

Data provider:

[✓] Always use this selection


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .\

Log on to the server

Authentication: Windows Authentication

User name:

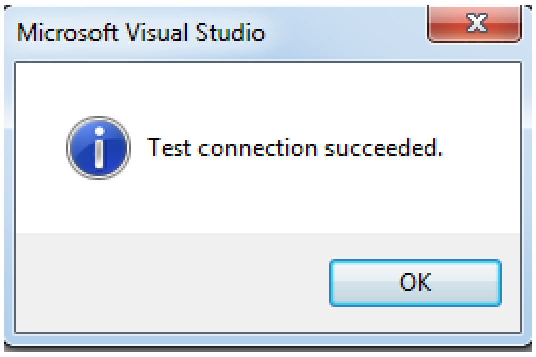
Password:

Save my password

Connect to a database

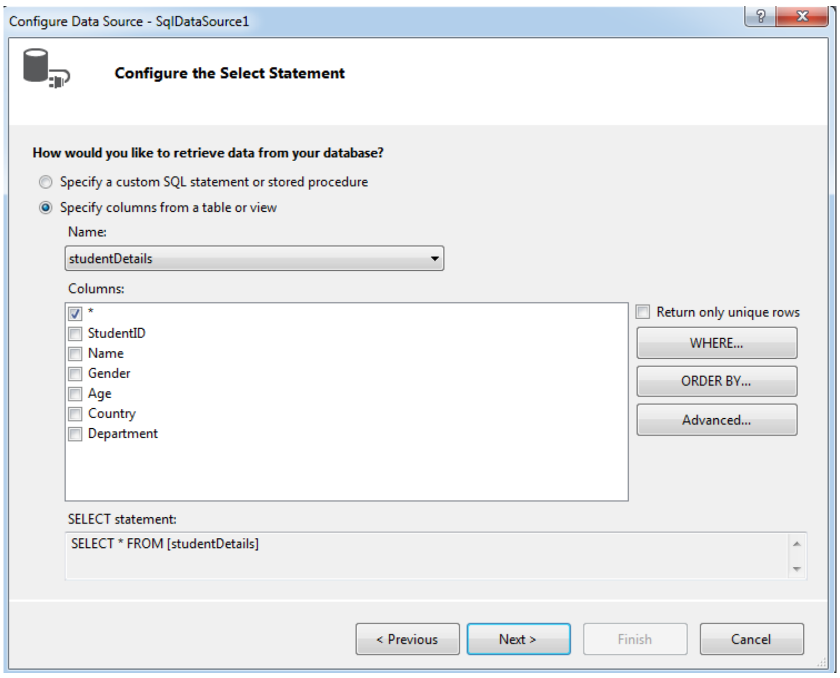
(●) Select or enter a database name:

- loginDB
- master
- model
- msdb
- ReportServer
- ReportServerTempDB
- StudentDB
- tempdb


Microsoft Visual Studio

Test connection succeeded.

(4) Once the connection is set up, you may save it for further use. At the next step, you are asked to configure the select statement:


Configure Data Source - SqlDataSource1

Configure the Select Statement

How would you like to retrieve data from your database?

( ) Specify a custom SQL statement or stored procedure

(●) Specify columns from a table or view

Name: studentDetails

Columns:

- [✓] *
- [ ] StudentID
- [ ] Name
- [ ] Gender
- [ ] Age
- [ ] Country
- [ ] Department

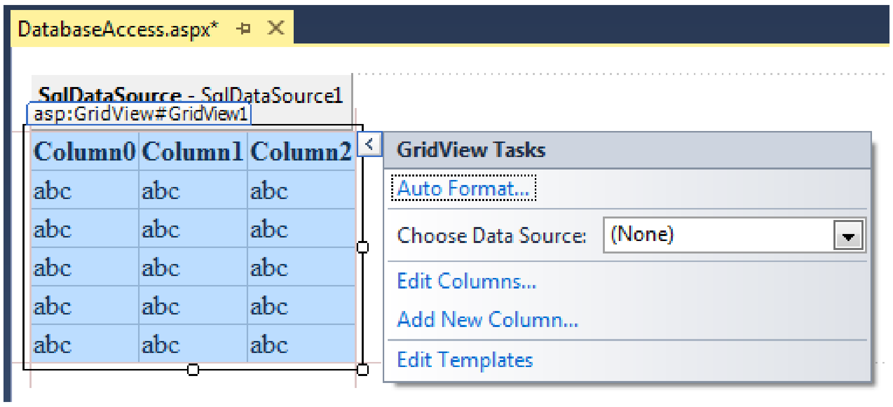
[ ] Return only unique rows

SELECT statement:

```
SELECT * FROM [studentDetails]
```

(5) Select the columns and click next to complete the steps. Observe the WHERE, ORDER BY, and the Advanced buttons. These buttons allow you to provide the where clause, order by clause, and specify the insert, update, and delete commands of SQL respectively. This way, you can manipulate the data.

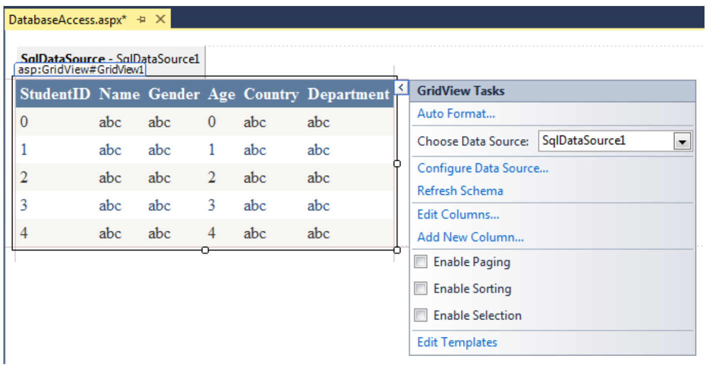
(6) Add a GridView control on the form. Choose the data source and format the control using AutoFormat option.



Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Choose Data Source: (None)

(7) After this, the formatted GridView control displays the column headings, and the application is ready to execute.

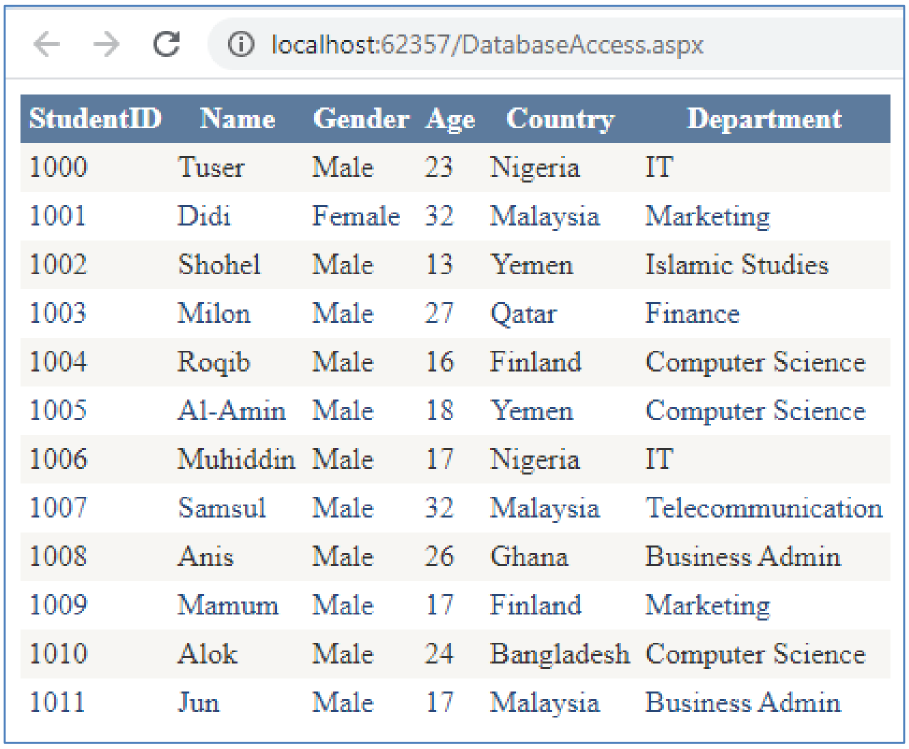


StudentID	Name	Gender	Age	Country	Department
0	abc	abc	0	abc	abc
1	abc	abc	1	abc	abc
2	abc	abc	2	abc	abc
3	abc	abc	3	abc	abc
4	abc	abc	4	abc	abc

Choose Data Source: SqlDataSource1

☐ Enable Paging
☐ Enable Sorting
☐ Enable Selection

(8) Finally execute the application.



StudentID	Name	Gender	Age	Country	Department
1000	Tuser	Male	23	Nigeria	IT
1001	Didi	Female	32	Malaysia	Marketing
1002	Shohel	Male	13	Yemen	Islamic Studies
1003	Milon	Male	27	Qatar	Finance
1004	Roqib	Male	16	Finland	Computer Science
1005	Al-Amin	Male	18	Yemen	Computer Science
1006	Muhiddin	Male	17	Nigeria	IT
1007	Samsul	Male	32	Malaysia	Telecommunication
1008	Anis	Male	26	Ghana	Business Admin
1009	Mamum	Male	17	Finland	Marketing
1010	Alok	Male	24	Bangladesh	Computer Science
1011	Jun	Male	17	Malaysia	Business Admin

The content file code is as given:

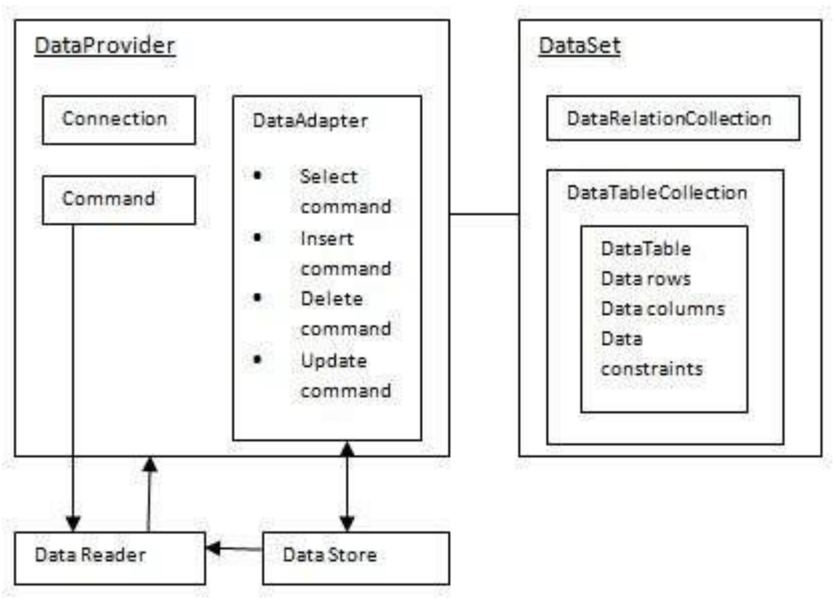
```
<%@            Page            Language="C#"            AutoEventWireup="true"
CodeBehind="DatabaseAccess.aspx.cs"  Inherits="LUC_Web_Sample2.DatabaseAccess"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:SqlDataSource          ID="SqlDataSource1"          runat="server"
ConnectionString="<%$     ConnectionStrings:StudentDBConnectionString     %>"
SelectCommand="SELECT * FROM [studentDetails]"></asp:SqlDataSource>
    </div>
        <div>
            <asp:GridView            ID="GridView1"             runat="server"
AutoGenerateColumns="False"      CellPadding="4"      DataKeyNames="StudentID"
DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None">
                <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
                <Columns>
                    <asp:BoundField                           DataField="StudentID"
HeaderText="StudentID" ReadOnly="True" SortExpression="StudentID" />
                    <asp:BoundField     DataField="Name"     HeaderText="Name"
SortExpression="Name" />
                    <asp:BoundField   DataField="Gender"   HeaderText="Gender"
SortExpression="Gender" />
                    <asp:BoundField      DataField="Age"      HeaderText="Age"
SortExpression="Age" />
                    <asp:BoundField  DataField="Country"  HeaderText="Country"
SortExpression="Country" />
                    <asp:BoundField                         DataField="Department"
HeaderText="Department" SortExpression="Department" />
                </Columns>
                <EditRowStyle BackColor="#999999" />
                <FooterStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
                <HeaderStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
                <PagerStyle BackColor="#284775" ForeColor="White"
HorizontalAlign="Center" />
                <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
                <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True"
ForeColor="#333333" />
                <SortedAscendingCellStyle BackColor="#E9E7E2" />
                <SortedAscendingHeaderStyle BackColor="#506C8C" />
                <SortedDescendingCellStyle BackColor="#FFFDF8" />
                <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
            </asp:GridView>
        </div>
    </form>
</body>
</html>
```

Working with ActiveX Data Objects (ADO.NET)

ActiveX Data Objects (ADO) is an application program interface from Microsoft that lets a programmer writing applications get access to a relational or non-relational database from both Microsoft and other database providers.

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

The following figure shows the ADO.NET objects at a glance:



The DataSet Class

The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects. The DataRelation objects represent the relationship between two tables.

Following table shows some important properties of the DataSet class:

Properties	**Description**
CaseSensitive	Indicates whether string comparisons within the data tables are case-sensitive.
Container	Gets the container for the component.
DataSetName	Gets or sets the name of the current data set.
DefaultViewManager	Returns a view of data in the data set.
DesignMode	Indicates whether the component is currently in design mode.
EnforceConstraints	Indicates whether constraint rules are followed when attempting any update operation.
Events	Gets the list of event handlers that are attached to this component.
ExtendedProperties	Gets the collection of customized user information associated with the DataSet.
HasErrors	Indicates if there are any errors.
IsInitialized	Indicates whether the DataSet is initialized.
Locale	Gets or sets the locale information used to compare strings within the table.
Namespace	Gets or sets the namespace of the DataSet.
Prefix	Gets or sets an XML prefix that aliases the namespace of the DataSet.
Relations	Returns the collection of DataRelation objects.
Tables	Returns the collection of DataTable objects.

The following table shows some important methods of the DataSet class:

Methods	**Description**
AcceptChanges	Accepts all changes made since the DataSet was loaded or this method was called.
BeginInit	Begins the initialization of the DataSet. The initialization occurs at run time.
Clear	Clears data.
Clone	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.

Copy	Copies both structure and data.
CreateDataReader()	Returns a DataTableReader with one result set per DataTable, in the same sequence as the tables appear in the Tables collection.
CreateDataReader(DataTable[])	Returns a DataTableReader with one result set per DataTable.
EndInit	Ends the initialization of the data set.
Equals(Object)	Determines whether the specified Object is equal to the current Object.
Finalize	Free resources and perform other cleanups.
GetChanges	Returns a copy of the DataSet with all changes made since it was loaded or the AcceptChanges method was called.
GetChanges(DataRowState)	Gets a copy of DataSet with all changes made since it was loaded or the AcceptChanges method was called, filtered by DataRowState.
GetDataSetSchema	Gets a copy of XmlSchemaSet for the DataSet.
GetObjectData	Populates a serialization information object with the data needed to serialize the DataSet.
GetType	Gets the type of the current instance.
GetXML	Returns the XML representation of the data.
GetXMLSchema	Returns the XSD schema for the XML representation of the data.
HasChanges()	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows.
HasChanges(DataRowState)	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows, filtered by DataRowState.
IsBinarySerialized	Inspects the format of the serialized representation of the DataSet.
Load(IDataReader, LoadOption, DataTable[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of DataTable instances to supply the schema and namespace information.
Load(IDataReader, LoadOption, String[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of strings to supply the names for the tables within the DataSet.

Merge()	Merges the data with data from another DataSet. This method has different overloaded forms.
ReadXML()	Reads an XML schema and data into the DataSet. This method has different overloaded forms.
ReadXMLSchema(0)	Reads an XML schema into the DataSet. This method has different overloaded forms.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
WriteXML()	Writes an XML schema and data from the DataSet. This method has different overloaded forms.
WriteXMLSchema()	Writes the structure of the DataSet as an XML schema. This method has different overloaded forms.

The DataTable Class

The DataTable class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the PrimaryKey property:

Properties	**Description**
ChildRelations	Returns the collection of child relationship.
Columns	Returns the Columns collection.
Constraints	Returns the Constraints collection.
DataSet	Returns the parent DataSet.
DefaultView	Returns a view of the table.
ParentRelations	Returns the ParentRelations collection.
PrimaryKey	Gets or sets an array of columns as the primary key for the table.
Rows	Returns the Rows collection.

The following table shows some important methods of the DataTable class:

Methods	**Description**
AcceptChanges	Commits all changes since the last AcceptChanges.
Clear	Clears all data from the table.
GetChanges	Returns a copy of the DataTable with all changes made since the AcceptChanges method was called.
GetErrors	Returns an array of rows with errors.

ImportRows	Copies a new row into the table.
LoadDataRow	Finds and updates a specific row, or creates a new one, if not found any.
Merge	Merges the table with another DataTable.
NewRow	Creates a new DataRow.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
Reset	Resets the table to its original state.
Select	Returns an array of DataRow objects.

The DataRow Class

The DataRow object represents a row in a table. It has the following important properties:

Properties	**Description**
HasErrors	Indicates if there are any errors.
Items	Gets or sets the data stored in a specific column.
ItemArrays	Gets or sets all the values for the row.
Table	Returns the parent table.

The following table shows some important methods of the DataRow class:

Methods	**Description**
AcceptChanges	Accepts all changes made since this method was called.
BeginEdit	Begins edit operation.
CancelEdit	Cancels edit operation.
Delete	Deletes the DataRow.
EndEdit	Ends the edit operation.
GetChildRows	Gets the child rows of this row.
GetParentRow	Gets the parent row.
GetParentRows	Gets parent rows of DataRow object.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.

The DataAdapter Object

The DataAdapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.

The DataReader Object

The DataReader object is an alternative to the DataSet and DataAdapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

DbCommand and DbConnection Objects

The DbConnection object represents a connection to the data source. The connection could be shared among different command objects.

The DbCommand object represents the command or a stored procedure sent to the database from retrieving or manipulating data.

Example

So far, we have used tables and databases already existing in our computer. In this example, we will create a table, add column, rows and data into it and display the table using a GridView object.

The source file code is as given:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="gridViewExample.aspx.cs"
Inherits="LUC_Web_Sample2.gridViewExample" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:GridView ID="GridView1" runat="server" CellPadding="4"
ForeColor="#333333" GridLines="None">
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
            <EditRowStyle BackColor="#999999" />
            <FooterStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
            <HeaderStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
            <PagerStyle BackColor="#284775" ForeColor="White"
HorizontalAlign="Center" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True"
ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#E9E7E2" />
            <SortedAscendingHeaderStyle BackColor="#506C8C" />
            <SortedDescendingCellStyle BackColor="#FFFDF8" />
            <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
        </asp:GridView>
    </div>
    </form>
</body>
</html>
```

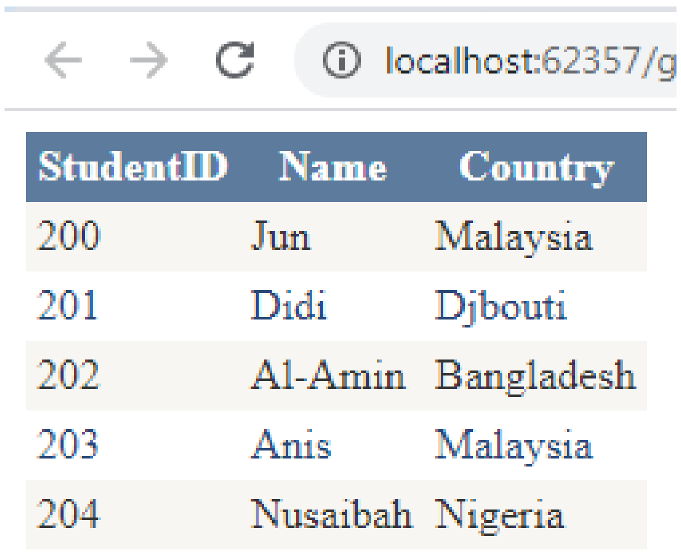
The code behind file is as given:

```
using System;
using System.Data;


namespace LUC_Web_Sample2
{
    public partial class gridViewExample : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                DataSet ds = CreateDataSet();
                GridView1.DataSource = ds.Tables["Student"];
                GridView1.DataBind();
            }

        }
        private DataSet CreateDataSet()
        {
            //creating a DataSet object for tables
            DataSet dataset = new DataSet();

            // creating the student table
            DataTable Students = CreateStudentTable();
            dataset.Tables.Add(Students);
            return dataset;
        }

        private DataTable CreateStudentTable()
        {
            DataTable Students = new DataTable("Student");

            // adding columns
            AddNewColumn(Students, "System.Int32", "StudentID");
            AddNewColumn(Students, "System.String", "Name");
            AddNewColumn(Students, "System.String", "Country");

            // adding rows
            AddNewRow(Students, 200, "Jun", "Malaysia");
            AddNewRow(Students, 201, "Didi", "Djbouti");
            AddNewRow(Students, 202, "Al-Amin", "Bangladesh");
            AddNewRow(Students, 203, "Anis", "Malaysia");
            AddNewRow(Students, 204, "Nusaibah", "Nigeria");

            return Students;
        }

        private void AddNewColumn(DataTable table, string columnType, string
columnName)
        {
            DataColumn column = table.Columns.Add(columnName,
Type.GetType(columnType));
        }
        //adding data into the table
        private void AddNewRow(DataTable table, int id, string name, string
city)
        {
            DataRow newrow = table.NewRow();
            newrow["StudentID"] = id;
            newrow["Name"] = name;
            newrow["Country"] = city;
            table.Rows.Add(newrow);
        }
    }

}
```

When you execute the program, observe the following:

- The application first creates a data set and binds it with the grid view control using the DataBind() method of the GridView control.
- The Createdataset() method is a user defined function, which creates a new DataSet object and then calls another user defined method CreateStudentTable() to create the table and add it to the Tables collection of the data set.
- The CreateStudentTable() method calls the user defined methods AddNewColumn() and AddNewRow() to create the columns and rows of the table as well as to add data to the rows.

When the page is executed, it returns the rows of the table as shown:



StudentID	Name	Country
200	Jun	Malaysia
201	Didi	Djbouti
202	Al-Amin	Bangladesh
203	Anis	Malaysia
204	Nusaibah	Nigeria

Introduction to File Uploading

ASP.NET has two controls that allow users to upload files to the web server. Once the server receives the posted file data, the application can save it, check it, or ignore it. The following controls allow the file uploading:

- HtmlInputFile - an HTML server control
- FileUpload - and ASP.NET web control

Both controls allow file uploading, but the FileUpload control automatically sets the encoding of the form, whereas the HtmlInputFile does not do so.

In this lecture, we use the FileUpload control. The FileUpload control allows the user to browse for and select the file to be uploaded, providing a browse button and a text box for entering the filename.

Once, the user has entered the filename in the text box by typing the name or browsing, the SaveAs method of the FileUpload control can be called to save the file to the disk.

The basic syntax of FileUpload is:

```
<asp:FileUpload ID= "Uploader" runat = "server" />
```

The FileUpload class is derived from the WebControl class, and inherits all its members. Apart from those, the FileUpload class has the following read-only properties:

Properties	**Description**
FileBytes	Returns an array of the bytes in a file to be uploaded.
FileContent	Returns the stream object pointing to the file to be uploaded.
FileName	Returns the name of the file to be uploaded.
HasFile	Specifies whether the control has a file to upload.
PostedFile	Returns a reference to the uploaded file.

The posted file is encapsulated in an object of type HttpPostedFile, which could be accessed through the PostedFile property of the FileUpload class.

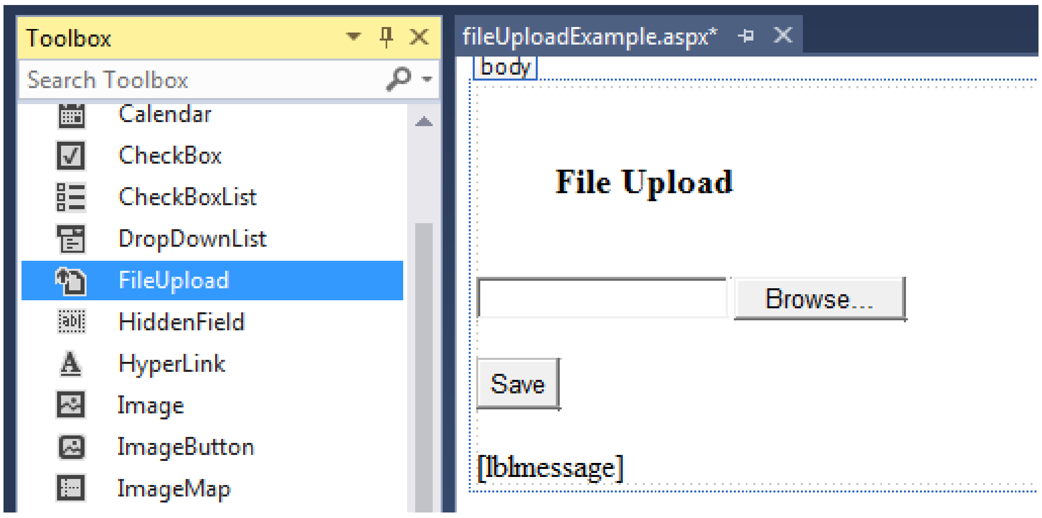
The HttpPostedFile class has the following frequently used properties:

Properties	**Description**
ContentLength	Returns the size of the uploaded file in bytes.
ContentType	Returns the MIME type of the uploaded file.
FileName	Returns the full filename.
InputStream	Returns a stream object pointing to the uploaded file.

Example

The following example demonstrates the FileUpload control and its properties. The form has a FileUpload control along with a save button and a label control for displaying the file name, file type, and file length.

In the design view, the form looks as follows:



The content file code is as given:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="fileUploadExample.aspx.cs"
Inherits="LUC_Web_Sample2.fileUploadExample" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
      <br />
         <h3>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; File Upload</h3>
         <br />
        <asp:FileUpload ID="FileUpload1" runat="server" />
        <br />
        <br />
        <asp:Button ID="btnsave" runat="server" Text="Save"
OnClick="btnsave_Click" />
        <br />
        <br />
        <asp:Label ID="lblmessage" runat="server"></asp:Label>
    </div>
    </form>
</body>
</html>
```

The code behind the save button is as given:

```
using System;
using System.Text;

namespace LUC_Web_Sample2
{
    public partial class fileUploadExample : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnsave_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder();

            if (FileUpload1.HasFile)
            {
                try
                {
                    sb.AppendFormat(" Uploading file: {0}", 
FileUpload1.FileName);

                    //saving the file
                    FileUpload1.SaveAs(@"C:\Users\Munir\Desktop\UploadedData\" 
+ FileUpload1.FileName);

                    //Showing the file information
                    sb.AppendFormat("<br/> Save As: {0}", 
FileUpload1.PostedFile.FileName);
                    sb.AppendFormat("<br/> File type: {0}", 
FileUpload1.PostedFile.ContentType);
                    sb.AppendFormat("<br/> File length: {0}", 
FileUpload1.PostedFile.ContentLength);
                    sb.AppendFormat("<br/> File name: {0}", 
FileUpload1.PostedFile.FileName);

                }
                catch (Exception ex)
                {
                    sb.Append("<br/> Error <br/>");
                    sb.AppendFormat("Unable to save file <br/> {0}", 
ex.Message);
                }
            }
            lblmessage.Text = sb.ToString();
        }
    }
}
```

Note the following:

- The StringBuilder class is derived from System.Text namespace, so it needs to be included.
- The try and catch blocks are used for catching errors, and display the error message.

Working with Ad Rotator

The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.

The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

The basic syntax of adding an AdRotator is as follows:

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"  Target = 
"_blank" />
```

Before going into the details of the AdRotator control and its properties, let us look into the construction of the advertisement file.

The Advertisement File

The advertisement file is an XML file, which contains the information about the advertisements to be displayed.

Extensible Markup Language (XML) is a W3C standard for text document markup. It is a text-based markup language that enables you to store data in a structured format by using meaningful tags. The term 'extensible' implies that you can extend your ability to describe a document by defining meaningful tags for the application.

XML is not a language in itself, like HTML, but a set of rules for creating new markup languages. It is a meta-markup language. It allows developers to create custom tag sets for special uses. It structures, stores, and transports the information.

Following is an example of XML file:

```
<BOOK>
   <NAME> Learn XML </NAME>
   <AUTHOR> Samuel Peterson </AUTHOR>
   <PUBLISHER> NSS Publications </PUBLISHER>
   <PRICE> $30.00</PRICE>
</BOOK>
```

Like all XML files, the advertisement file needs to be a structured text file with well-defined tags delineating the data. The following are the standard XML elements that are commonly used in the advertisement file:

Element	**Description**
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

Apart from these tags, customs tags with custom attributes could also be included. The following code illustrates an advertisement file ads.xml:

```
<Advertisements>
   <Ad>
      <ImageUrl>rose1.jpg</ImageUrl>
      <NavigateUrl>http://www.1800flowers.com</NavigateUrl>
      <AlternateText>
         Order flowers, roses, gifts and more
      </AlternateText>
      <Impressions>20</Impressions>
      <Keyword>flowers</Keyword>
   </Ad>

   <Ad>
      <ImageUrl>rose2.jpg</ImageUrl>
      <NavigateUrl>http://www.babybouquets.com.au</NavigateUrl>
      <AlternateText>Order roses and flowers</AlternateText>
      <Impressions>20</Impressions>
      <Keyword>gifts</Keyword>
   </Ad>

   <Ad>
      <ImageUrl>rose3.jpg</ImageUrl>
      <NavigateUrl>http://www.flowers2moscow.com</NavigateUrl>
      <AlternateText>Send flowers to Russia</AlternateText>
      <Impressions>20</Impressions>
      <Keyword>russia</Keyword>
   </Ad>

   <Ad>
      <ImageUrl>rose4.jpg</ImageUrl>
      <NavigateUrl>http://www.edibleblooms.com</NavigateUrl>
      <AlternateText>Edible Blooms</AlternateText>
      <Impressions>20</Impressions>
      <Keyword>gifts</Keyword>
   </Ad>
</Advertisements>
```

Properties and Events of the AdRotator Class

The AdRotator class is derived from the WebControl class and inherits its properties. Apart from those, the AdRotator class has the following properties:

Properties	**Description**
AdvertisementFile	The path to the advertisement file.
AlternateTextFeild	The element name of the field where alternate text is provided. The default value is AlternateText.
DataMember	The name of the specific list of data to be bound when advertisement file is not used.
DataSource	Control from where it would retrieve data.
DataSourceID	Id of the control from where it would retrieve data.
Font	Specifies the font properties associated with the advertisement banner control.
ImageUrlField	The element name of the field where the URL for the image is provided. The default value is ImageUrl.
KeywordFilter	For displaying the keyword based ads only.

NavigateUrlField	The element name of the field where the URL to navigate to is provided. The default value is NavigateUrl.
Target	The browser window or frame that displays the content of the page linked.
UniqueID	Obtains the unique, hierarchically qualified identifier for the AdRotator control.

Following are the important events of the AdRotator class:

Events	**Description**
AdCreated	It is raised once per round trip to the server after creation of the control, but before the page is rendered
DataBinding	Occurs when the server control binds to a data source.
DataBound	Occurs after the server control binds to a data source.
Disposed	Occurs when a server control is released from memory, which is the last stage of the server control lifecycle when an ASP.NET page is requested
Init	Occurs when the server control is initialized, which is the first step in its lifecycle.
Load	Occurs when the server control is loaded into the Page object.
PreRender	Occurs after the Control object is loaded but prior to rendering.
Unload	Occurs when the server control is unloaded from memory.

Working with AdRotator Control

Create a new web page and place an AdRotator control on it.

```
<form id="form1" runat="server">
   <div>
      <asp:AdRotator    ID="AdRotator1"    runat="server"    AdvertisementFile
="~/ads.xml" onadcreated="AdRotator1_AdCreated" />
   </div>
</form>
```

The ads.xml file and the image files should be located in the root directory of the web site.

Try to execute the above application and observe that each time the page is reloaded, the ad is changed.

Working with Calendar Control

The calendar control is a functionally rich web control, which provides the following capabilities:

- Displaying one month at a time
- Selecting a day, a week or a month
- Selecting a range of days
- Moving from month to month
- Controlling the display of the days programmatically

The basic syntax of a calendar control is:

```
<asp:Calender ID = "Calendar1" runat = "server">

</asp:Calender>
```

Properties and Events of the Calendar Control

The calendar control has many properties and events, using which you can customize the actions and display of the control. The following table provides some important properties of the Calendar control:

Properties	**Description**
Caption	Gets or sets the caption for the calendar control.
CaptionAlign	Gets or sets the alignment for the caption.
CellPadding	Gets or sets the number of spaces between the data and the cell border.
CellSpacing	Gets or sets the space between cells.
DayHeaderStyle	Gets the style properties for the section that displays the day of the week.
DayNameFormat	Gets or sets format of days of the week.
DayStyle	Gets the style properties for the days in the displayed month.
FirstDayOfWeek	Gets or sets the day of week to display in the first column.
NextMonthText	Gets or sets the text for next month navigation control. The default value is >.
NextPrevFormat	Gets or sets the format of the next and previous month navigation control.
OtherMonthDayStyle	Gets the style properties for the days on the Calendar control that are not in the displayed month.
PrevMonthText	Gets or sets the text for previous month navigation control. The default value is <.
SelectedDate	Gets or sets the selected date.
SelectedDates	Gets a collection of DateTime objects representing the selected dates.
SelectedDayStyle	Gets the style properties for the selected dates.

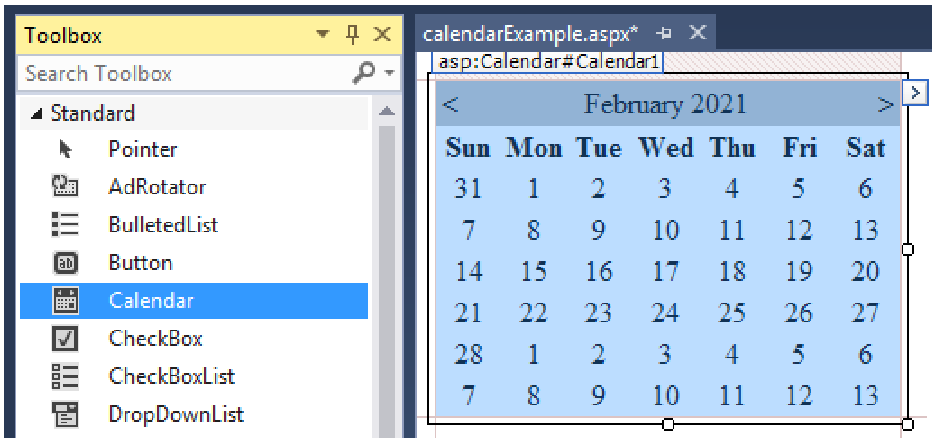
SelectionMode	Gets or sets the selection mode that specifies whether the user can select a single day, a week or an entire month.
SelectMonthText	Gets or sets the text for the month selection element in the selector column.
SelectorStyle	Gets the style properties for the week and month selector column.
SelectWeekText	Gets or sets the text displayed for the week selection element in the selector column.
ShowDayHeader	Gets or sets the value indicating whether the heading for the days of the week is displayed.
ShowGridLines	Gets or sets the value indicating whether the gridlines would be shown.
ShowNextPrevMonth	Gets or sets a value indicating whether next and previous month navigation elements are shown in the title section.
ShowTitle	Gets or sets a value indicating whether the title section is displayed.
TitleFormat	Gets or sets the format for the title section.
Titlestyle	Get the style properties of the title heading for the Calendar control.
TodayDayStyle	Gets the style properties for today's date on the Calendar control.
TodaysDate	Gets or sets the value for today's date.
UseAccessibleHeader	Gets or sets a value that indicates whether to render the table header <th> HTML element for the day headers instead of the table data <td> HTML element.
VisibleDate	Gets or sets the date that specifies the month to display.
WeekendDayStyle	Gets the style properties for the weekend dates on the Calendar control.

The Calendar control has the following three most important events that allow the developers to program the calendar control. They are:

Events	**Description**
SelectionChanged	It is raised when a day, a week or an entire month is selected.
DayRender	It is raised when each data cell of the calendar control is rendered.
VisibleMonthChanged	It is raised when user changes a month.

Working with the Calendar Control

Putting a bare-bone calendar control without any code behind file provides a workable calendar to a site, which shows the months and days of the year. It also allows navigation to next and previous months.



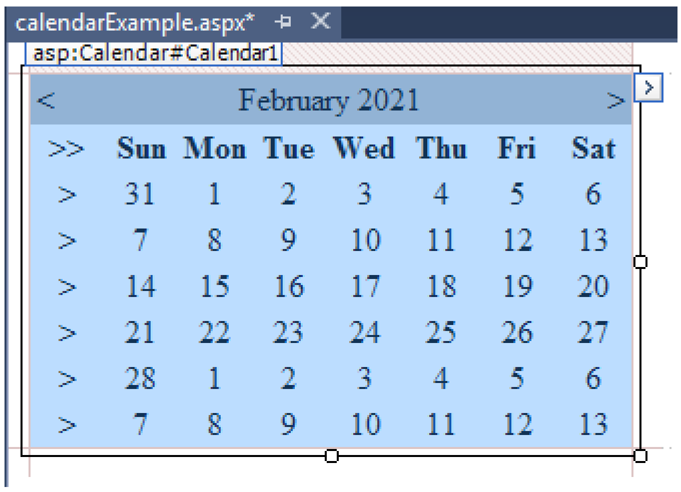
Calendar controls allow the users to select a single day, a week, or an entire month. This is done by using the SelectionMode property. This property has the following values:

Properties	**Description**
Day	To select a single day.
DayWeek	To select a single day or an entire week.
DayWeekMonth	To select a single day, a week, or an entire month.
None	Nothing can be selected.

The syntax for selecting days:

```
<asp:Calender ID = "Calendar1" runat = "server" SelectionMode="DayWeekMonth">

</asp:Calender>
```

When the selection mode is set to the value DayWeekMonth, an extra column with the > symbol appears for selecting the week, and a >> symbol appears to the left of the days name for selecting the month.



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
>	31	1	2	3	4	5	6
>	7	8	9	10	11	12	13
>	14	15	16	17	18	19	20
>	21	22	23	24	25	26	27
>	28	1	2	3	4	5	6
>	7	8	9	10	11	12	13

Example

The following example demonstrates selecting a date and displays the date in a label:

The content file code is as follows:

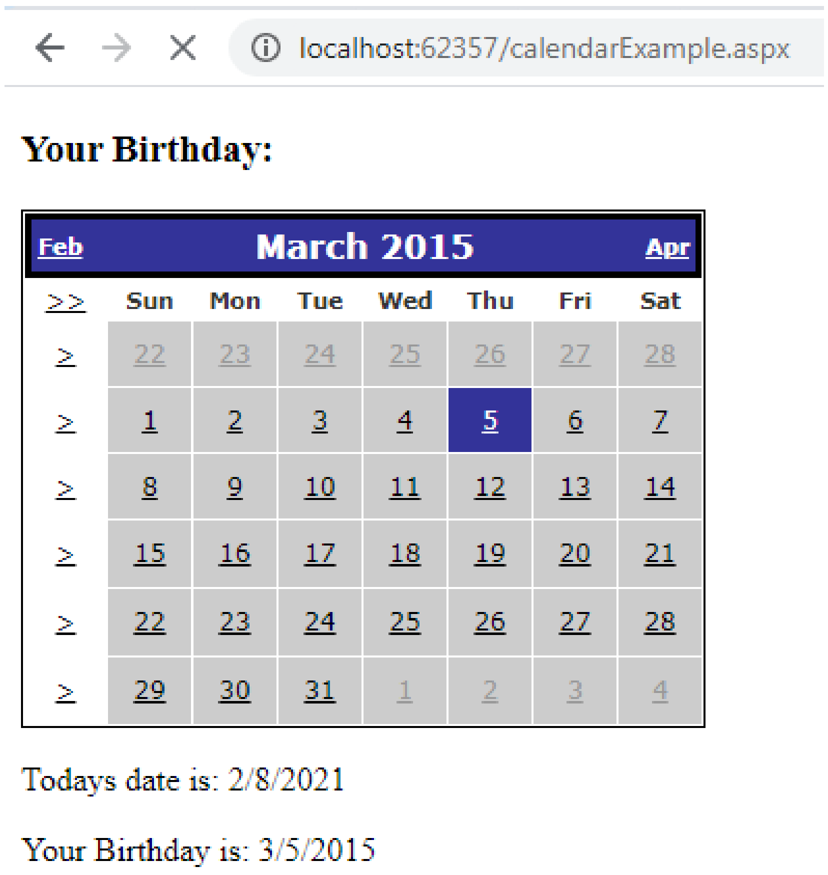
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="calendarExample.aspx.cs"
Inherits="LUC_Web_Sample2.calendarExample" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
         <h3> Your Birthday:</h3>
        <asp:Calendar ID="Calendar1" runat="server" BackColor="White"
BorderColor="Black" Font-Names="Verdana" Font-Size="9pt" ForeColor="Black"
Height="250px" NextPrevFormat="ShortMonth" SelectionMode="DayWeekMonth"
Width="330px" BorderStyle="Solid" CellSpacing="1"
OnSelectionChanged="Calendar1_SelectionChanged">
            <DayHeaderStyle Font-Bold="True" Font-Size="8pt"
ForeColor="#333333" Height="8pt" />
            <DayStyle BackColor="#CCCCCC" />
            <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White"
/>
            <OtherMonthDayStyle ForeColor="#999999" />
            <SelectedDayStyle BackColor="#333399" ForeColor="White" />
            <TitleStyle BackColor="#333399" BorderStyle="Solid" Font-
Bold="True" Font-Size="12pt" ForeColor="White" Height="12pt" />
            <TodayDayStyle BackColor="#999999" ForeColor="White" />
        </asp:Calendar>
    </div>
        <p>Todays date is:
            <asp:Label ID="lblday" runat="server"></asp:Label>
        </p>
        <p>
            Your Birthday is:
            <asp:Label ID="lblbday" runat="server"></asp:Label>
        </p>
    </form>
</body>
```

```
</html>
```

The event handler for the event SelectionChanged:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace LUC_Web_Sample2
{
    public partial class calendarExample : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Calendar1_SelectionChanged(object sender, EventArgs e)
        {
            lblday.Text = Calendar1.TodaysDate.ToShortDateString();
            lblbday.Text = Calendar1.SelectedDate.ToShortDateString();
        }
    }
}
```

When the file is run, it should produce the following output:



localhost:62357/calendarExample.aspx

Your Birthday:

Feb			March 2015				Apr
>>	Sun	Mon	Tue	Wed	Thu	Fri	Sat
>	22	23	24	25	26	27	28
>	1	2	3	4	5	6	7
>	8	9	10	11	12	13	14
>	15	16	17	18	19	20	21
>	22	23	24	25	26	27	28
>	29	30	31	1	2	3	4

Todays date is: 2/8/2021

Your Birthday is: 3/5/2015

ACTIVITY

1. What is ADO.NET?
2. Highlight any 5 methods of ADO.NET DataSet class
3. Highlight any 5 properties of ADO.NET DataTable class
4. Highlight any 5 methods of ADO.NET DataRow class
5. Describe DataAdapter, DataReader, DbCommand and DbConnection Objects with appropriate example.
6. Differentiate between HtmlInputFile and FileUpload
7. What is the function of AdRotator and how is it implemented?
8. Highlight any 5 standard XML elements that are commonly used in the advertisement file.
9. Highlight any 5 properties of the Calendar control in ASP.Net
10. What is the syntax for selecting days on an ASP.net calendar control?

KEYWORDS

- XML
- ADO.NET
- DataSet
- AdRotator
- FileUpload

SUMMARY

This chapter presents the following topics:

- Database Access in ASP.Net
- Retrieve and display data
- Working with ActiveX Data Objects (ADO.NET)
- The DataSet Class
- The DataTable Class
- The DataRow Class
- The DataAdapter Object
- The DataReader Object
- DbCommand and DbConnection Objects
- Introduction to File Uploading
- Working with Ad Rotator
- Properties and Events of the AdRotator Class
- Calendar Control