



AMIT ACADEMY
for Computer Education



Subject Title: WEB DEVELOPMENT USING ASP.NET

Subject Code: BCA 501

**Unit 4 Working with Database and State
Management**

Prepared by Ankit Rami(AR) Any Query

Contact – +91 8460467193

Email – ankitramiblog@gmail.com

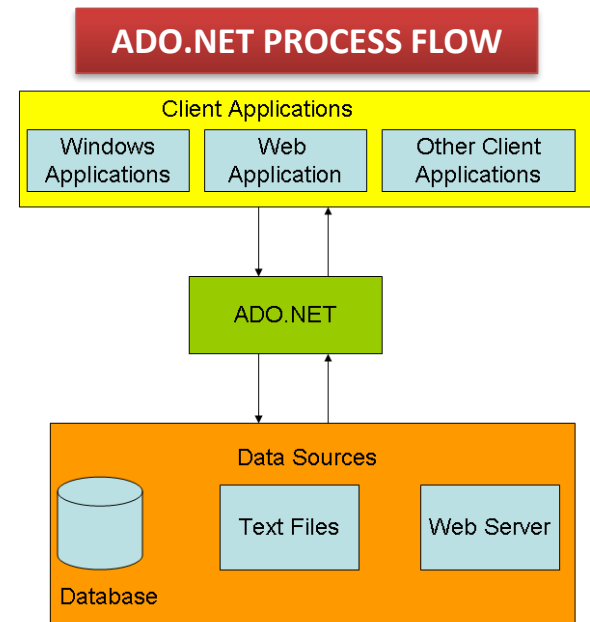
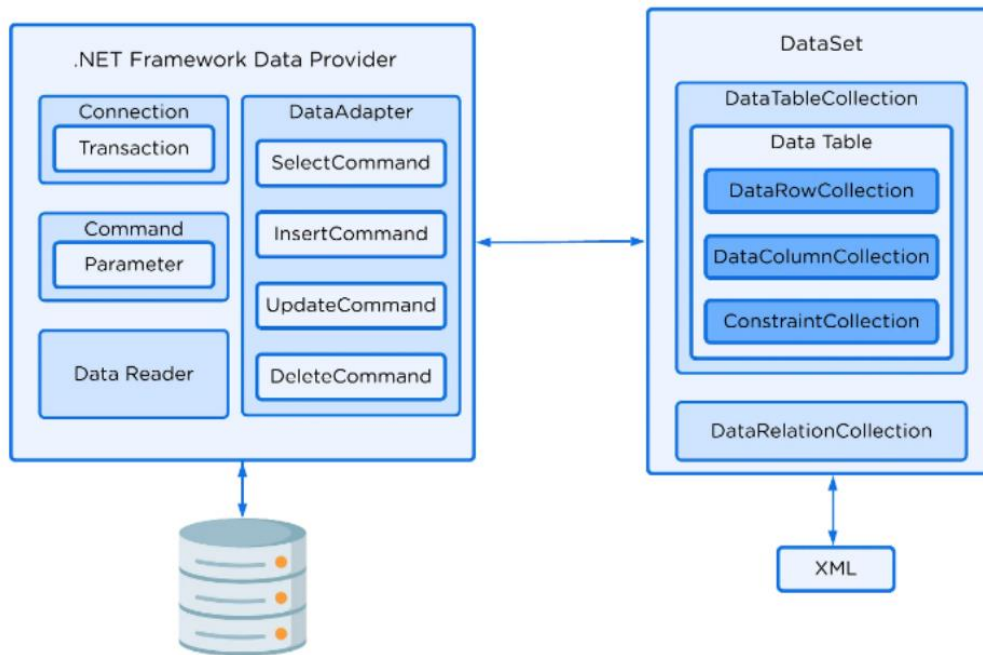
Overview of ADO.NET

- ✓ ADO was introduced in 1996 by Microsoft as its component of MDAC (Microsoft Data Access Components).
- ✓ It is based on COM (Component Object Modelling). ADO with other components of MDAC serves as a framework for client applications to access data stores.
- ✓ It removes necessitate to know implementation of database and lowers complexity of handling low level code requires for dealing with data.
- ✓ ADO.NET is an advanced database technology from Microsoft .NET Framework which provides communication between application system and database server.
- ✓ It is a component of the .NET Framework that is designed to work on disconnected model to access data from data store.
- ✓ ADO.NET is a module of .NET Framework which is used to establish connection between application and data sources.
- ✓ Data sources can be such as SQL Server and XML.
- ✓ ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.
- ✓ All the ADO.NET classes are located into **System.Data.dll** and integrated with XML classes located into **System.Xml.dll**.
- ✓ The following are a few of the .NET applications that use ADO.NET to connect to a database, execute commands and retrieve data from the database.
 1. ASP.NET Web Applications
 2. Console Applications
 3. Windows Applications.

Features of ADO.Net

- ✓ It provides some in built classes to connect databases like mysql, sql, oracle etc...
- ✓ In built classes to do operations like insert, update, delete and select.
- ✓ Integration with XML
- ✓ Disconnected Data architecture for better performance
- ✓ Persistence Ignorance
- ✓ Lazy Loading
- ✓ Self Tracking Entities
- ✓ Bulk copying of data from a data source to another data source is a new feature added to ADO.NET 2.0.
- ✓ Batch update can provide a huge improvement in the performance by making just one round trip to the server for multiple batch updates, instead of several trips if the database server supports the batch update feature.
- ✓ Data Paging concept you can simply call this method with start data indexing in per page.
- ✓ **Interoperability** - We know that XML documents are text-based formats. So, one can edit and edit XML documents using standard text-editing tools. ADO.NET uses XML in all data exchanges and for internal representation of data.
- ✓ **Maintainability** – ADO.NET is built around the idea of separation of data logic and user interface. It means that we can create our application in independent layers.
- ✓ **Programmability** (Typed Programming) – It is a programming style in which user words are used to construct statements or evaluate expressions. For example: If we want to select the “Marks” column from “Kawal” from the “Student” table, the following is the way to do so:
- ✓ **Performance** – It uses disconnected data architecture which is easy to scale as it reduces the load on the database. Everything is handled on the client-side, so it improves performance.
- ✓ **Scalability** – It means meeting the needs of the growing number of clients, which degrading performance. As it uses disconnected data access, applications do not retain database lock connections for a longer time. Thus, it accommodates scalability by encouraging programmers to conserve limited resources and allow users to access data simultaneously.

ADO.NET Architecture



- ✓ ADO.NET offers a link between the front-end interfaces and the back-end databases. All data access activities are encapsulated as ADO.NET objects, which are then interacted with by controls to show data while concealing the specifics of data transfer.
- ✓ The ADO.NET architecture comprises six important components.
 1. Connection
 2. Command
 3. DataReader
 4. DataAdapter
 5. DataSet
 6. DataView

ADO.NET Architecture

Connection Object

- ✓ ADO.NET's Connection object should be our initial port of call. It is through connections that ADO.NET is connected to data sources. Between a Data Adapter and a Data Source, there is a Connection object. When a Command object is attached, it may perform SQL commands that can be used to get data from a data source, update, or remove data.
- ✓ An important part of a transaction is the creation of the connection. The good functionality for working with transactions, like commit and rollback, is kept in transaction objects.

Command Object

- ✓ Executing SQL statements and data structures is possible with the Command object. A DataSet or a Data Reader object returns data from SQL statements. SELECT, INSERT, UPDATE, and DELETE Database queries are used to obtain, add, and remove data. These statements may be found in a DataAdapter created using VS .NET IDE.

Data Adapter Object

- ✓ The DataAdapter's function is implicit in its title: it conducts the operations required to transfer data from the server's data source to the database maintained by the DataSet. Data Adapter: We may define instructions for retrieving and updating data using the Data Adapter. To connect a data source to the Dataset, use the Data Adapter. The Data Adapter is aware of the DataSet and the appropriate methods for populating it. It is also known to the Adapter that a link to the data source has been established.

ADO.NET Architecture

Data Reader Object

- ✓ One of the two techniques offered by ADO.NET, the Data Reader object, is used to read information from the data store. As we will recall, the Data Reader object offers high efficiency, a read-only, forward-only mechanism for retrieving data as a stream of data from a data repository while maintaining a connection to the data source. The Data Reader is constrained yet extremely efficient.

Dataset Object

- ✓ The ADO.NET Data Reader object is one of the two techniques provided by ADO.NET. This object is utilized to obtain data from the data store. As we will recall, the Data Reader object offers a read-only, forward-only, high-performance mechanism for retrieving information from the data store as a data stream while maintaining a connection with the data source. This framework is read-only because it can only read data written after it has been read. The Data Reader has several limitations, but it's very well tuned.

ADO.NET Data Source Controls

- ✓ A data source control interacts with the data-bound controls and hides the complex data binding processes. These are the tools that provide data to the data bound controls and support execution of operations like insertions, deletions, sorting, and updates
- ✓ Each data source control wraps a particular data provider-relational databases, XML documents, or custom classes and helps in:
 - 1.Managing connection
 - 2.Selecting data
 - 3.Managing presentation aspects like paging, caching, etc.
 - 4.Manipulating data
- ✓ There are many data source controls available in ASP.NET for accessing data from SQL Server, from ODBC or OLE DB servers, from XML files, and from business objects.
- ✓ Based on type of data, these controls could be divided into two categories:
 - 1.Hierarchical data source controls
 - 2.Table-based data source controls
- ✓ The data source controls used for hierarchical data are:
 - 1.XMLDataSource - It allows binding to XML files and strings with or without schema information.
 - 2.SiteMapDataSource - It allows binding to a provider that supplies site map information.

ADO.NET Data Source Controls

- ✓ SqlDataSource control is a data source control provided in ASP.NET to connect to database providers such as SQL, OLEDB, ODBC, and Oracle. This control only establishes a connection with the data source; it does not display data on the web page. The data is displayed by binding the SqlDataSource control to a data-bound control such as a GridView or DataList. These data-bound controls, in turn, display the data on the web page. The SqlDataSource control also supports editing, updating, deleting, and sorting of the records retrieved from the database. This support to manipulate the data in the data source can be implemented in data-bound controls without writing any code. In other words, a SqlDataSource control allows you to access databases without creating a Connection, Command, or a Data Reader object.
- ✓ Adding SqlDataSource Control - A SqlDataSource control is added to the webform using markup or by dragging it from the Toolbox and dropping it on the web form. The code below shows the markup to create a SqlDataSource control.

```
<asp:SqlDataSource ID="sqlDsSuppliers" runat="server">
</asp:SqlDataSource>
```

- ✓ Configuring SqlDataSource Control - The basic configuration of the SqlDataSource control involves setting two attributes, namely ConnectionString and SelectCommand. The ConnectionString attribute specifies the connection string to connect to a particular database. The SelectCommand attribute specifies the select statement to retrieve the records from a database table. Source code uses these attributes to connect to the Northwind database located on the SQL Server instance, SQLEXPRESS, installed on the system 10.2.1.51.

```
<asp:SqlDataSource ID="sqlDsSuppliers" runat="server" ConnectionString="Data
Source=10.2.1.51\SQLEXPRESS;Initial Catalog=Northwind; Integrated Security=True"
SelectCommand="select * from NWSuppliers;">
</asp:SqlDataSource>
```


ADO.NET SqlConnection Class

- ✓ It is used to establish an open connection to the SQL Server database. It is a sealed class so that cannot be inherited. SqlConnection class uses SqlDataAdapter and SqlCommand classes together to increase performance when connecting to a Microsoft SQL Server database.
- ✓ Connection does not close explicitly even it goes out of scope. Therefore, you must explicitly close the connection by calling Close() method.

SqlConnection Constructors

Constructors	Description
SqlConnection()	It is used to initializes a new instance of the SqlConnection class.
SqlConnection(String)0	It is used to initialize a new instance of the SqlConnection class and takes connection string as an argument.
SqlConnection(String, SqlCredential)	It is used to initialize a new instance of the SqlConnection class that takes two parameters. First is connection string and second is sql credentials.

SqlConnection Methods

Method	Description
BeginTransaction()	It is used to start a database transaction.
ChangeDatabase(String)	It is used to change the current database for an open SqlConnection.
ChangePassword(String, String)	It changes the SQL Server password for the user indicated in the connection string.
Close()	It is used to close the connection to the database.
CreateCommand()	It enlists in the specified transaction as a distributed transaction.
GetSchema()	It returns schema information for the data source of this SqlConnection.
Open()	It is used to open a database connection.
ResetStatistics()	It resets all values if statistics gathering is enabled.

ADO.NET SqlConnection Class

SqlConnection Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class CRUD : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection cn = new SqlConnection(@"Data
        Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\KSV\Desktop\ADO_
        Connection\App_Data\Database.mdf;Integrated Security=True;Connect
        Timeout=30");
        cn.Open();
        Response.Write("Database Connected");
        cn.Close();
    }
}

```

ADO.NET SqlCommand Class

- ✓ This class is used to store and execute SQL statement for SQL Server database. It is a sealed class so that cannot be inherited.

SqlCommand Constructors

Constructor	Description
SqlCommand()	It is used to initialize a new instance of the SqlCommand class.
SqlCommand(String)	It is used to initialize a new instance of the SqlCommand class with a string parameter.
SqlCommand(String, SqlConnection)	It is used to initialize a new instance of the SqlCommand class. It takes two parameters, first is query string and second is connection string.
SqlCommand(String, SqlConnection, SqlTransaction)	It is used to initialize a new instance of the SqlCommand class. It takes three parameters query, connection and transaction string respectively.
SqlCommand(String, SqlConnection, SqlTransaction, SqlCommandColumnEncryptionSetting)	It Initializes a new instance of the SqlCommand class with specified command text, connection, transaction, and encryption setting.

SqlCommand Methods

Method	Description
BeginExecuteNonQuery()	It is used to Initiate the asynchronous execution of the SQL statement described by this SqlCommand.
Cancel()	It tries to cancel the execution of a SqlCommand.
Clone()	It creates a new SqlCommand object that is a copy of the current instance.
CreateParameter()	It creates a new instance of a SqlParameter object.
ExecuteReader()	It is used to send the CommandText to the Connection and builds a SqlDataReader.
ExecuteXmlReader()	It is used to send the CommandText to the Connection and builds an XmlReader object.
ExecuteScalar()	It executes the query and returns the first column of the first row in the result set. Additional columns or rows are ignored.
Prepare()	It is used to create a prepared version of the command by using the instance of SQL Server.
ResetCommandTimeout()	It is used to reset the CommandTimeout property to its default value.

ADO.NET SqlCommand Class

SqlCommand Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class CRUD : System.Web.UI.Page
{
    SqlConnection cn = new SqlConnection(@"Data
Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\KSV\Desktop\ADO_Connection\App_Data\Database.mdf;Int
egrated Security=True;ConnectTimeout=30");
    protected void Button1_Click(object sender, EventArgs e)
    {
        cn.Open();
        SqlCommand cmd = new SqlCommand("INSERT INTO student(sname,scon)Values('" +
TextBox1.Text + "', '" + TextBox2.Text + "')",cn);
        cmd.ExecuteNonQuery();
        Response.Write("<script>alert('Data Insert Successfully')</script>");
        cn.Close();
    }
}

```

ADO.NET SqlDataAdapter Class

- ✓ The DataAdapter works as a bridge between a DataSet and a data source to retrieve data. DataAdapter is a class that represents a set of SQL commands and a database connection. It can be used to fill the DataSet and update the data source.

SQLDataAdapter Constructors

Constructors	Description
DataAdapter()	It is used to initialize a new instance of a DataAdapter class.
DataAdapter(DataAdapter)	It is used to initializes a new instance of a DataAdapter class from an existing object of the same type.

SQLDataAdapter Methods

Method	Description
CloneInternals()	It is used to create a copy of this instance of DataAdapter.
Dispose(Boolean)	It is used to release the unmanaged resources used by the DataAdapter.
Fill(DataSet)	It is used to add rows in the DataSet to match those in the data source.
FillSchema(DataSet, SchemaType, String, IDataReader)	It is used to add a DataTable to the specified DataSet.
GetFillParameters()	It is used to get the parameters set by the user when executing an SQL SELECT statement.
ResetFillLoadOption()	It is used to reset FillLoadOption to its default state.
ShouldSerializeAcceptChangesDuringFill()	It determines whether the AcceptChangesDuringFill property should be persisted or not.
ShouldSerializeFillLoadOption()	It determines whether the FillLoadOption property should be persisted or not.
ShouldSerializeTableMappings()	It determines whether one or more DataTableMapping objects exist or not.
Update(DataSet)	It is used to call the respective INSERT, UPDATE, or DELETE statements.

ADO.NET SqlDataAdapter Class

SQLDataAdapter Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class CRUD : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection cn = new SqlConnection(@"Data
            Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\KSV\Desktop\ADO_
            Connection\App_Data\Database.mdf;Integrated Security=True;Connect
            Timeout=30");
        cn.Open();
        SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM student", cn);
        DataSet ds = new DataSet();
        da.Fill(ds, "student");
        GridView1.DataSource = ds.Tables["student"].DefaultView;
        GridView1.DataBind();
        cn.Close();
    }
}

```

ADO.NET DataSet Class

- ✓ It is a collection of data tables that contain the data. It is used to fetch data without interacting with a Data Source that's why, it also known as disconnected data access method. It is an inmemory data store that can hold more than one table at the same time. We can use DataRelation object to relate these tables. The DataSet can also be used to read and write data as XML document.
- ✓ ADO.NET provides a DataSet class that can be used to create DataSet object. It contains constructors and methods to perform data related operations..

SQLDataSet Constructors

Constructor	Description
DataSet()	It is used to initialize a new instance of the DataSet class.
DataSet(String)	It is used to initialize a new instance of a DataSet class with the given name.
DataSet(SerializationInfo, StreamingContext)	It is used to initialize a new instance of a DataSet class that has the given serialization information and context.
DataSet(SerializationInfo, StreamingContext, Boolean)	It is used to initialize a new instance of the DataSet class.

ADO.NET DataSet Class

DataSet Methods

Method	Description
BeginInit()	It is used to begin the initialization of a DataSet that is used on a form.
Clear()	It is used to clear the DataSet of any data by removing all rows in all tables.
Clone()	It is used to copy the structure of the DataSet.
Copy()	It is used to copy both the structure and data for this DataSet.
CreateDataReader(DataTable[])	It returns a DataTableReader with one result set per DataTable.
CreateDataReader()	It returns a DataTableReader with one result set per DataTable.
EndInit()	It ends the initialization of a DataSet that is used on a form.
GetXml()	It returns the XML representation of the data stored in the DataSet.
GetXmlSchema()	It returns the XML Schema for the XML representation of the data stored in the DataSet.
Load(IDataReader, LoadOption, DataTable[])	It is used to fill a DataSet with values from a data source using the supplied IDataReader.
Merge(DataSet)	It is used to merge a specified DataSet and its schema into the current DataSet.
Merge(DataTable)	It is used to merge a specified DataTable and its schema into the current DataSet.
ReadXml(XmlReader, XmlReadMode)	It is used to read XML schema and data into the DataSet using the specified XmlReader and XmlReadMode.
Reset()	It is used to clear all tables and removes all relations, foreign constraints, and tables from the DataSet.
WriteXml(XmlWriter, XmlWriteMode)	It is used to write the current data and optionally the schema for the DataSet using the specified XmlWriter and XmlWriteMode.

ADO.NET DataSet Class

DataSet Properties

Properties	Description
CaseSensitive	It is used to check whether DataTable objects are case-sensitive or not.
DataSetName	It is used to get or set name of the current DataSet.
DefaultViewManager	It is used to get a custom view of the data contained in the DataSet to allow filtering and searching.
HasErrors	It is used to check whether there are errors in any of the DataTable objects within this DataSet.
IsInitialized	It is used to check whether the DataSet is initialized or not.
Locale	It is used to get or set the locale information used to compare strings within the table.
Namespace	It is used to get or set the namespace of the DataSet.
Site	It is used to get or set an ISite for the DataSet.
Tables	It is used to get the collection of tables contained in the DataSet.

ADO.NET SQLDataSet Class

SQLDataAdapter Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class CRUD : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection cn = new SqlConnection(@"Data
            Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\KSV\Desktop\ADO_
            Connection\App_Data\Database.mdf;Integrated Security=True;Connect
            Timeout=30");
        cn.Open();
        SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM student", cn);
        DataSet ds = new DataSet();
        da.Fill(ds, "student");
        GridView1.DataSource = ds.Tables["student"].DefaultView;
        GridView1.DataBind();
        cn.Close();
    }
}

```

ADO.NET SqlDataReader Class

- ✓ This class is used to read data from SQL Server database. It reads data in forward-only stream of rows from a SQL Server database. it is sealed class so that cannot be inherited. It inherits DbDataReader class and implements IDisposable interface.

SQLDataReader Method

Method	Description
Close()	It is used to closes the SqlDataReader object.
GetBoolean(Int32)	It is used to get the value of the specified column as a Boolean.
GetByte(Int32)	It is used to get the value of the specified column as a byte.
GetChar(Int32)	It is used to get the value of the specified column as a single character.
GetDateTime(Int32)	It is used to get the value of the specified column as a DateTime object.
GetDecimal(Int32)	It is used to get the value of the specified column as a Decimal object.
GetDouble(Int32)	It is used to get the value of the specified column as a double-precision floating point number.
GetFloat(Int32)	It is used to get the value of the specified column as a single-precision floating point number.
GetName(Int32)	It is used to get the name of the specified column.
GetSchemaTable()	It is used to get a DataTable that describes the column metadata of the SqlDataReader.
GetValue(Int32)	It is used to get the value of the specified column in its native format.
GetValues(Object[])	It is used to populate an array of objects with the column values of the current row.
NextResult()	It is used to get the next result, when reading the results of SQL statements.
Read()	It is used to read record from the SQL Server database.

ADO.NET SqlDataReader Class

SqlDataReader Properties

Property	Description
Connection	It is used to get the SqlConnection associated with the SqlDataReader.
Depth	It is used to get a value that indicates the depth of nesting for the current row.
FieldCount	It is used to get the number of columns in the current row.
HasRows	It is used to get a value that indicates whether the SqlDataReader contains one or more rows.
IsClosed	It is used to retrieve a boolean value that indicates whether the specified SqlDataReader instance has been closed.
Item[String]	It is used to get the value of the specified column in its native format given the column name.
Item[Int32]	It is used to get the value of the specified column in its native format given the column ordinal.
RecordsAffected	It is used to get the number of rows changed, inserted or deleted by execution of the Transact-SQL statement.
VisibleFieldCount	It is used to get the number of fields in the SqlDataReader that are not hidden.

ADO.NET SqlDataReader Class

SQLDataReader Code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class CRUD : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection cn = new SqlConnection(@"Data
            Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\KSV\Desktop\ADO_
            Connection\App_Data\Database.mdf;Integrated Security=True;Connect
            Timeout=30");
        cn.Open();
        SqlCommand cmd = new SqlCommand("select * from bca where aun='" + TextBox4.Text + "' and
apw='" + TextBox5.Text + "'", cn);
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.Read()==true)
        {
            Response.Write("<script>alert('Login Successfully')</script>");
        }
        else{
            Response.Write("<script>alert('Login Fail')</script>");
        }
        cn.Close();
    }
}

```

ADO.NET DataTable Class

- ✓ DataTable represents relational data into tabular form. ADO.NET provides a DataTable class to create and use data table independently. It can also be used with DataSet also. Initially, when we create DataTable, it does not have table schema. We can create table schema by adding columns and constraints to the table. After defining table schema, we can add rows to the table.
- ✓ We must include System.Data namespace before creating DataTable.

SQLDataTable Constructors

Constructors	Description
DataTable()	It is used to initialize a new instance of the DataTable class with no arguments.
DataTable(String)	It is used to initialize a new instance of the DataTable class with the specified table name.
DataTable(SerializationInfo, StreamingContext)	It is used to initialize a new instance of the DataTable class with the SerializationInfo and the StreamingContext.
DataTable(String, String)	It is used to initialize a new instance of the DataTable class using the specified table name and namespace.

ADO.NET DataTable Class

SQLDataTable Method

Method	Description
AcceptChanges()	It is used to commit all the changes made to this table.
Clear()	It is used to clear the DataTable of all data.
Clone()	It is used to clone the structure of the DataTable.
Copy()	It is used to copy both the structure and data of the DataTable.
CreateDataReader()	It is used to returns a DataTableReader corresponding to the data within this DataTable.
CreateInstance()	It is used to create a new instance of DataTable.
GetRowType()	It is used to get the row type.
GetSchema()	It is used to get schema of the table.
ImportRow(DataRow)	It is used to copy a DataRow into a DataTable.
Load(IDataReader)	It is used to fill a DataTable with values from a data source using the supplied IDataReader.
Merge(DataTable, Boolean)	It is used to merge the specified DataTable with the current DataTable.
NewRow()	It is used to create a new DataRow with the same schema as the table.
Select()	It is used to get an array of all DataRow objects.
WriteXml(String)	It is used to write the current contents of the DataTable as XML using the specified file.

ADO.NET DataTable Class

SQLDataTable Properties

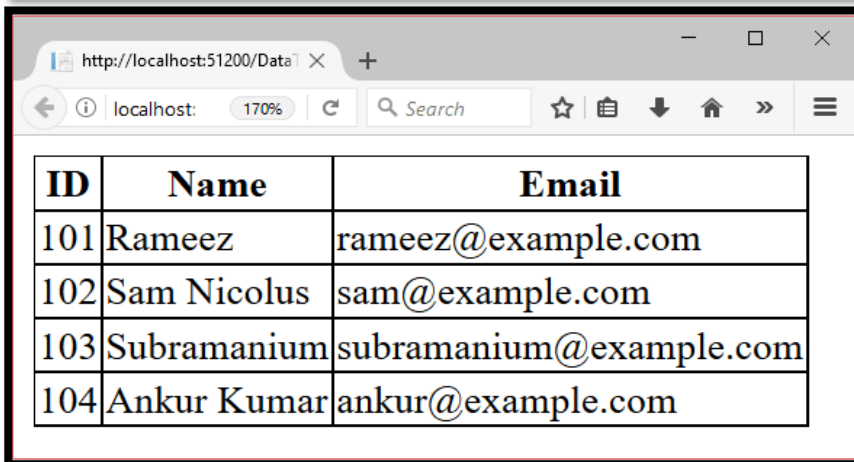
Property	Description
Columns	It is used to get the collection of columns that belong to this table.
Constraints	It is used to get the collection of constraints maintained by this table.
DataSet	It is used to get the DataSet to which this table belongs.
DefaultView	It is used to get a customized view of the table that may include a filtered view.
HasErrors	It is used to get a value indicating whether there are errors in any of the rows in the table of the DataSet.
MinimumCapacity	It is used to get or set the initial starting size for this table.
PrimaryKey	It is used to get or set an array of columns that function as primary keys for the data table.
Rows	It is used to get the collection of rows that belong to this table.
TableName	It is used to get or set the name of the DataTable.

ADO.NET SQLDataTable Class

SQLDataTable Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataTableForm.aspx.cs"
Inherits="DataTableDemo.DataTableForm" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title> </title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
    </div>
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
  </form>
</body>
</html>
```

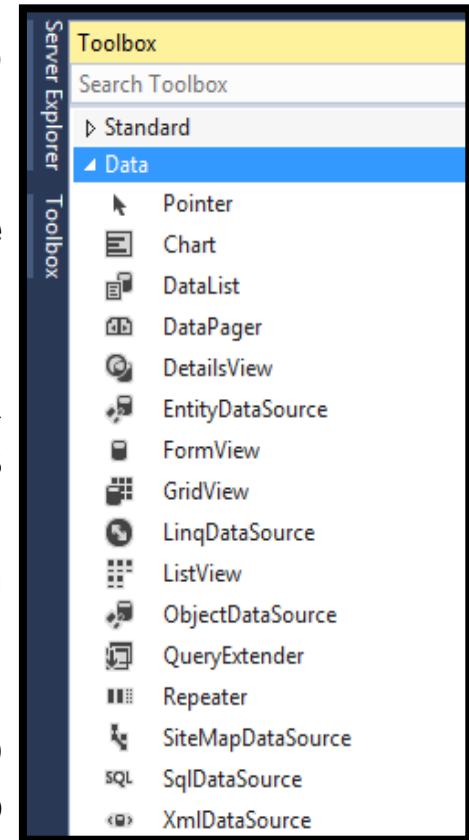
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace DataTableDemo
{
  public partial class DataTableForm : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      DataTable table = new DataTable();
      table.Columns.Add("ID");
      table.Columns.Add("Name");
      table.Columns.Add("Email");
      table.Rows.Add("101", "Rameez", "rameez@example.com");
      table.Rows.Add("102", "Sam Nicolus", "sam@example.com");
      table.Rows.Add("103", "Subramanium", "subramanium@example.com");
      table.Rows.Add("104", "Ankur Kumar", "ankur@example.com");
      GridView1.DataSource = table;
      GridView1.DataBind();
    }
  }
}
```



ID	Name	Email
101	Rameez	rameez@example.com
102	Sam Nicolus	sam@example.com
103	Subramanium	subramanium@example.com
104	Ankur Kumar	ankur@example.com

📢 Working with Data Bound Controls

- ✓ Databound controls are used to display data to the end-user within the web applications and using databound controls allows you to manipulate the data within the web applications very easily.
- ✓ Databound controls are bound to the DataSource property.
- ✓ Databound controls are composite controls that combine other ASP.NET Controls like Text boxes, Radio buttons, Buttons and so on.
- ✓ The data bound controls used in ASP.NET to display data in various forms and do various database activities such as Add, Edit, Update and Delete operations. The control makes the data more organized and presents the data in an efficient way for the viewers.
- ✓ ASP.NET provides a wide variety of rich controls that can be bound to data. Under the Data tab of the Visual Studio Toolbox, you can get several controls under the Data tab that could be used to display data from a data source, like a database or XML file.
- ✓ **1.GridView, 2.DataList, 3.DetailsView, 4.FormView, 5.ListView, 6.Chart**

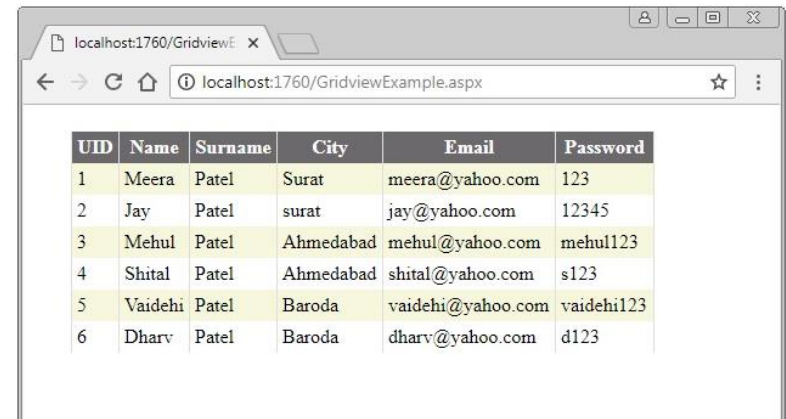
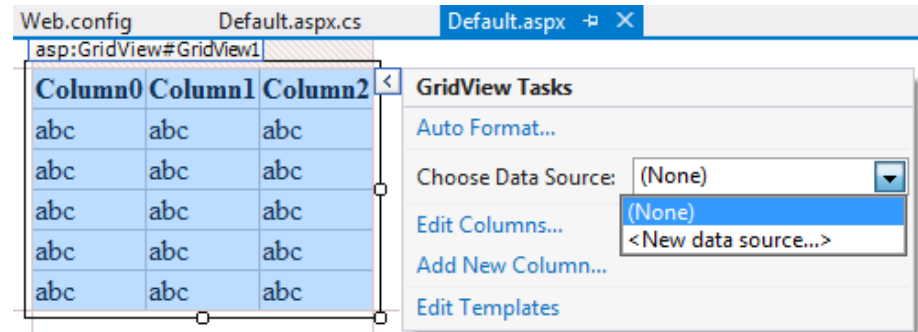


Working with Data Bound Controls

- ✓ Databound controls are used to display data to the end-user within the web applications and using databound controls allows you to manipulate the data within the web applications very easily.
- ✓ Databound controls are bound to the DataSource property.
- ✓ Databound controls are composite controls that combine other ASP.NET Controls like Text boxes, Radio buttons, Buttons and so on.
- ✓ The data bound controls used in ASP.NET to display data in various forms and do various database activities such as Add, Edit, Update and Delete operations. The control makes the data more organized and presents the data in an efficient way for the viewers.
- ✓ ASP.NET provides a wide variety of rich controls that can be bound to data. Under the Data tab of the Visual Studio Toolbox, you can get several controls under the Data tab that could be used to display data from a data source, like a database or XML file.
- ✓ **1.GridView 2.DataList 3.DetailsView,**
4.FormView 5.ListView 6.Chart

Working with Grid View Controls

- ✓ The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record.
- ✓ The GridView control supports the following features:
 - Binding to data source controls, such as SqlDataSource.
 - Built-in sort capabilities.
 - Built-in update and delete capabilities.
 - Built-in paging capabilities.
 - Built-in row selection capabilities.
 - Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
 - Multiple key fields.
 - Multiple data fields for the hyperlink columns.
 - Customizable appearance through themes and styles.



Working with Grid View Controls

Properties of GridView

AutoGenerateColumns:

This property allows you to automatically generate columns based on the data source.

AllowSorting:

This property allows you to enable or disable data sorting in the **GridView**.

AllowPaging:

This property allows you to enable or disable the paging of data in the **GridView**.

ShowHeader:

This property allows you to show or hide the header of the **GridView**.

ShowFooter:

This property allows you to show or hide the footer of the **GridView**.

EditIndex:

This property allows you to set the index of the row being edited in the **GridView**.

Working with Data List Controls

- ✓ DataList is a Databound control to display and manipulate data in a web application. It is a composite control that can combine other ASP.Net controls and it is present in the form. The DataList appearance is controlled by its template fields.
- ✓ The following template fields are supported by the DataList control:
- ✓ **Itemtemplate:** It specifies the Items present in the Datasource, it renders itself in the browser as many rows present in the data source collection.
- ✓ **EditItemTemplate:** Used to provide edit permissions to the user.
- ✓ **HeaderTemplate:** Used to display header text to the data source collection.
- ✓ **FooterTemplate:** Used to display footer text to the data source collection.
- ✓ **ItemStyle:** Used to apply styles to an ItemTemplate.
- ✓ **EditStyle:** Used to apply styles to an EditItemTemplate
- ✓ **HeaderStyle:** Used to apply styles to a HeaderTemplate
- ✓ **FooterStyle:** Used to apply styles to a FooterTemplate.

Working with Data List Controls

DataList Properties

Property Name	Description
RepeatDirection Horizontal Vertical(Default)	Used to set or get the Direction of items being render. Horizontal: Items will render in Horizontal. Vertical: Items will render in Vertical.
RepeatCoulmns	Used to set number of columns display in DataList. Return type is int data type
EditItemIndex	It is run-time property used to set an index value to change its template from ItemTemplate to EditItemTemplate. If it set to -1 no Items will change its Template

DataList Events

EventName	Description
OnEditCommand	It will be raised when user clicks on a button(present in DataList) whose CommandName is "Edit"
OnUpdateCommand	It will be raised when user clicks on a button(present in DataList) whose CommandName is "Update"
OnCancelCommand	It will be raised when user clicks on a button(present in DataList) whose CommandName is "Cancel"
OnDeleteCommand	It will be raised when user clicks on a button(present in DataList) whose CommandName is "Delete"

Working with Data List Controls

The image shows a Visual Studio IDE with an ASP.NET web application. The code in the background defines a DataList control with an ItemTemplate that displays student information. The rendered output in the browser shows a list of six students with alternating background colors for each row.

```

<title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:DataList ID="DataList1" runat="server" BackColor="white" BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" CellPadding="3" DataKeyField="sid" DataSourceID="SqlDataSource1">
        <AlternatingItemStyle BackColor="#CCCCCC" />
        <FooterStyle BackColor="#CCCCCC" />
        <HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="white" />
        <ItemTemplate>
          sid:
          <asp:Label ID="sidLabel" runat="server" Text="<%= Eval("sid") %>" />
          <br />
          sname:
          <asp:Label ID="snameLabel" runat="server" Text="<%= Eval("sname") %>" />
          <br />
          scon:
          <asp:Label ID="sconLabel" runat="server" Text="<%= Eval("scon") %>" />
          <br />
        </ItemTemplate>
        <SelectedItemStyle BackColor="#000099" Font-Bold="True" ForeColor="white" />
      </asp:DataList>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%= $%> ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM [student]"></asp:SqlDataSource>
    </div>
  </form>
</body>

```

Rendered Output:

sid: 1 sname: Ankit scon: 8460467193
sid: 2 sname: Deep scon: 8787457848
sid: 3 sname: Sagar scon: 9878457846
sid: 4 sname: Smit scon: 7845747485
sid: 6 sname: Jeet scon: 8978457845

Working with Details View Controls

- ✓ The DetailsView Control enable us to work with a single data item at a time. This control enable us to display, edit, insert, and delete data items such as database records. Furthermore, it also enable us to page forward and backward through a set of data items. The DetailsView control always renders each field in a separate HTML table row.

Displaying Data in DetailsView

- ✓ A DetailsView control renders an HTML table that displays the contents of a single database record. The DetailsView supports both declarative and programmatic databinding.
- ✓ ASP.Net DetailsView control with three BoundField columns and a SqlDataSource control.
- ✓ **AutoGenerateRows** : This property is used to disable automatically display of records from database. Generally when BoundField or TemplateField columns are used, this property needs to be set to false as by default the value is true.
- ✓ **AllowPaging**: In order to enable paging in the DetailsView control, the AllowPaging property needs to be set to true. By default paging is disabled in DetailsView control.
- ✓ **SqlDataSource** : The ID of the SqlDataSource control is set as DataSourceID for the DetailsView control. The ConnectionString property of SqlDataSource is set by specifying the name of the connection string setting in the Web.Config file.

Working with Details View Controls

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the source code for `ASP_DETAILSVIEW.aspx`. The code includes a `<asp:DetailsView>` control with various styling attributes and a `<asp:SqlDataSource>` control. The `SelectCommand` is `SELECT * FROM [student]`.
- Browser Window:** A preview window at `localhost:27446/ADO_Co...` shows the rendered output. It features a table with a dark green header and footer, and a white body. The table contains one row of data:

sid	1
sname	Ankit
scon	8460467193
1 2 3 4 5	
- Design View:** A small preview of the table structure is visible in the bottom-left corner of the IDE.
- IntelliTrace Panel:** Located on the right, it shows a message: "Streaming Video: Collecting and analyzing data in product..." and options to break or view settings.

Working with Form View Controls

- ✓ We use the FormView control to do anything that we also can do with the DetailsView control. Just as we can with the DetailsView control, we can use the FormView control to display, page, edit, insert, and delete database records. However, unlike the DetailsView control, the FormView control is entirely template driven. I end up using the FormView control much more than the DetailsView control. The FormView control provides us with more control over the layout of a form. Furthermore, adding validation controls to a FormView is easier than adding validation controls to a DetailsView control.

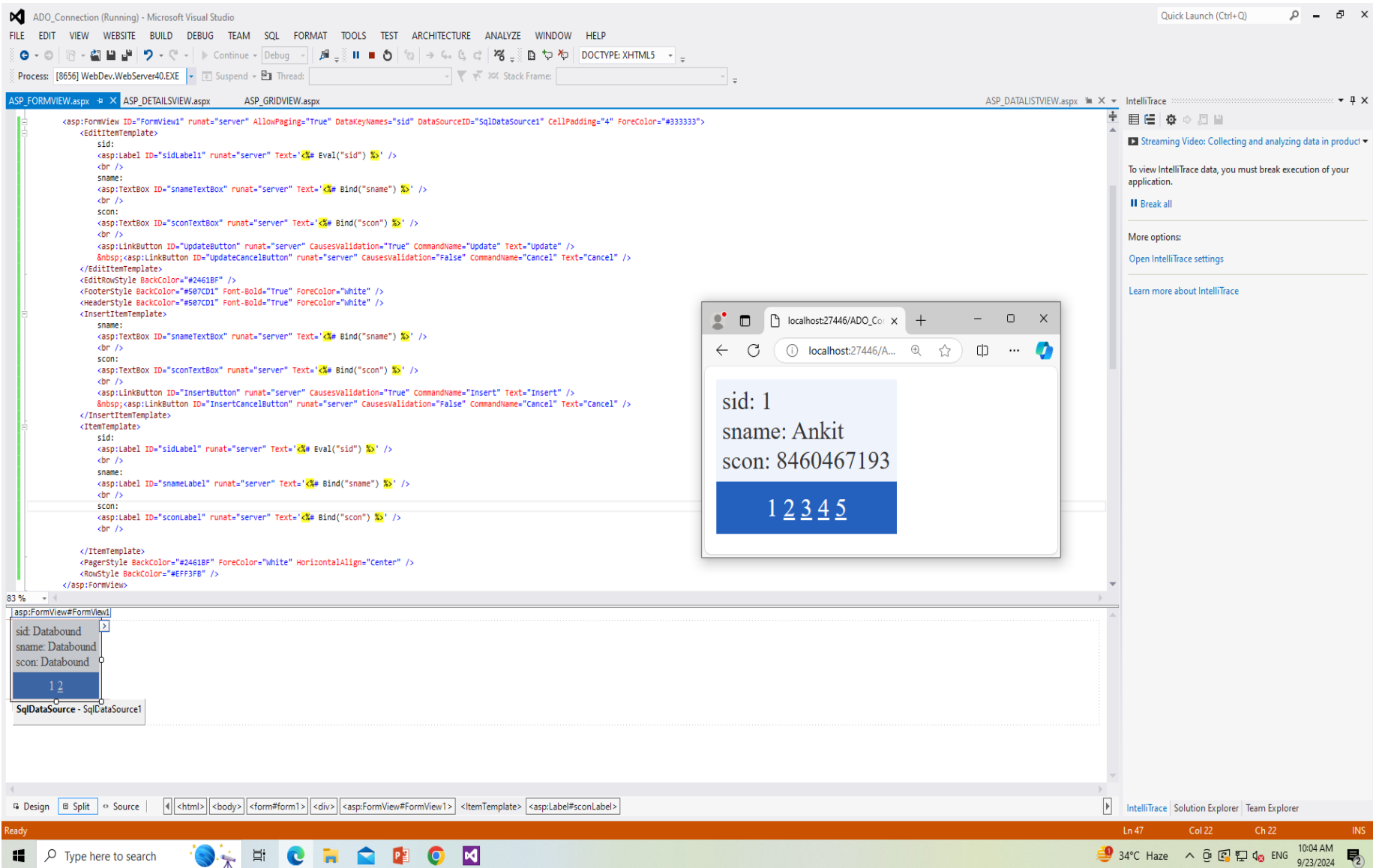
Displaying Data with FormView Control

- ✓ We can display a database record with the FormView control by using an ItemTemplate.

The FormView control supports the following features:

- ✓ Binding to data source controls, such as SqlDataSource and ObjectDataSource.
- ✓ Built-in inserting capabilities.
- ✓ Built-in updating and deleting capabilities.
- ✓ Built-in paging capabilities.

Working with Form View Controls



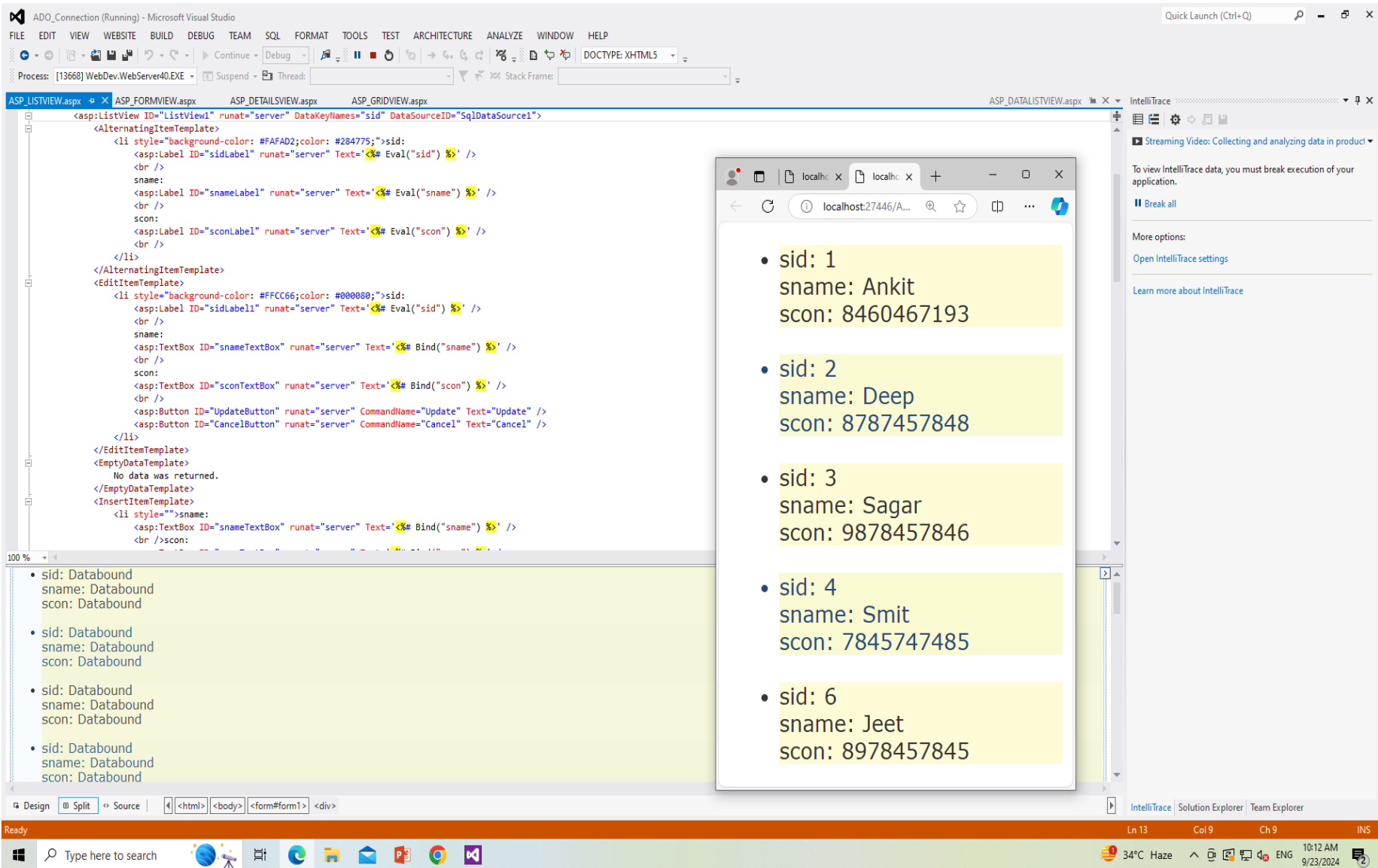
The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows ASP.NET code for `ASP_GRIDVIEW.aspx`. The code includes:
 - Page-level attributes: `AllowPaging="True"`, `DataKeyNames="sid"`, `DataSourceID="SqlDataSource1"`, `CellPadding="4"`, `ForeColor="#333333"`.
 - ItemTemplate: Contains a label for `sid`, a text box for `sname`, a text box for `scon`, and buttons for `Update` and `Cancel`.
 - EditItemTemplate: Contains a text box for `sname`, a text box for `scon`, and an `Insert` button.
 - ItemTemplate: Contains labels for `sid`, `sname`, and `scon`.
 - Page-level styling: `PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center"`, `RowStyle BackColor="#FFF3FB"`.
- Browser Preview:** A window at `localhost:27446/ADO_Co...` shows the rendered output:
 - Text: `sid: 1`
 - Text: `sname: Ankit`
 - Text: `scon: 8460467193`
 - Buttons: A blue button with the text `1 2 3 4 5`.
- IntelliTrace Panel:** Shows a message: "Streaming Video: Collecting and analyzing data in product..." and "To view IntelliTrace data, you must break execution of your application." with a "Break all" button.
- Design View:** A small preview at the bottom left shows a grid with columns for `sid`, `sname`, and `scon`, and a button labeled `1 2`. Below it is the `SqlDataSource - SqlDataSource1` control.
- Toolbox:** Shows the `asp:FormView#FormView1` control selected.

Working with List View Controls

- ✓ The ListView control displays columns and rows of data and allows sorting and paging. It is by far the most popular data display control, and is ideal for understanding how data display controls interact with data retrieval controls and code.
- ✓ The ListView has the following Templates:
- ✓ LayoutTemplate :- This determines the layout of the whole ListView. This section of ListView will have a Placeholder enclosed in HTML tags. An HTML Table has been placed inside the LayoutTemplate.
- ✓ ItemTemplate :- This contains the actual DataBound items which will be populated from Database, similar to the ItemTemplate of ASP.Net GridView or DataList controls. Inside the ItemTemplate, records will be populated from the database.
- ✓ GroupTemplate :- This plays the vital role in repeating the ListView items horizontally, and it is associated with the GroupItemCount property. This section of ListView will have a Placeholder enclosed in HTML tags. The GroupPlaceholder will display the contents from GroupTemplate.
- ✓ DataPager control :- Then there is an ASP.Net DataPager control placed inside the LayoutTemplate which will be used to display the pager. The DataPager control has been assigned with following properties:
- ✓ PagedControlID - It has been set with the ID of the ListView control.
- ✓ PageSize – It is used to set the number of records to display on the ListView. In this case it is 10.

Working with List View Controls



The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows ASP.NET code for a list view control. The code includes an `<asp:ListView ID="ListView1" runat="server" DataKeyNames="sid" DataSourceID="SqlDataSource1">` tag with an `<AlternatingItemTemplate>` and an `<EditItemTemplate>`. The `<AlternatingItemTemplate>` contains labels for `sid`, `sname`, and `scon`. The `<EditItemTemplate>` contains text boxes for `sname` and `scon`, and buttons for `Update` and `Cancel`.
- Browser Preview:** A window showing the rendered output of the list view control. It displays a list of items with the following data:
 - sid: 1, sname: Ankit, scon: 8460467193
 - sid: 2, sname: Deep, scon: 8787457848
 - sid: 3, sname: Sagar, scon: 9878457846
 - sid: 4, sname: Smit, scon: 7845747485
 - sid: 6, sname: Jeet, scon: 8978457845
- IntelliTrace:** A panel on the right side of the IDE showing streaming video and options to view IntelliTrace data.
- Taskbar:** Shows the Windows taskbar with the Start button, search bar, and various application icons.

Working with Chart View Controls

- ✓ **ASP.NET Chart Control** offers a comprehensive collection of 2D and 3D charts to meet the data visualization needs of your end users.
- ✓ The ASP.NET Chart Control provides flexible data binding options that allow you to bind to a table from a database or a collection created in code.
- ✓ The control also includes a built-in chart wizard that is invoked at design time, to assist in chart configuration.
- ✓ The ASP.NET Chart control can plot over 30 chart types ranging from line charts to specialized financial charts. Its rich feature set includes functionalities like data binding, multiple axes, trackball, drill-down operations, and zooming.
- ✓ All modern browsers and devices are supported. The chart is rendered as SVG in all the modern browsers and automatically rendered as VML on Internet Explorer 8 and below.
- ✓ Includes several data rendering optimizations to achieve the best possible performance when plotting large volumes of data as well as handling high frequency real-time data.

Chart Types

- ✓ The chart control includes functionality for plotting more than 30 chart types. Each chart type is easily configurable with built-in support for creating stunning visual effects.
- ✓ Line and area type charts for representing time-dependent data, showing trends in data at equal intervals.
- ✓ Column and bar type charts for comparing the frequency, count, total, or average of data in different categories. They are ideal for showing variations in the value of an item over time.
- ✓ Pie and pyramid type charts to represent data in proportions.
- ✓ Polar and radar charts to perform visual comparisons among several quantitative or qualitative aspects of a situation.
- ✓ Bubble and scatter charts to plot financial or scientific data.
- ✓ Waterfall chart for representing the cumulative effect of sequential positive or negative values.
- ✓ Candle and HiLo type charts for stock analysis.
- ✓ Range column and range area series to represent high and low values at a point.
- ✓ 3-D charts for pie, doughnut, column, and bar type series.

Working with Chart View Controls

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the ASP.NET code for `ASP_CHART.aspx`. The code registers the `System.Web.DataVisualization` assembly and uses `asp:Chart` and `asp:SqlDataSource` controls. The chart is configured with a data source and a series of bars.
- Design View:** Shows a preview of the rendered chart with 7 bars of varying heights.
- Browser Window:** Shows the rendered chart in a browser at `localhost:27446/ADO_Co...`. The chart displays data for five students: Ankit (1), Deep (2), Sagor (3), Smit (4), and Jeet (6).
- IntelliTrace:** A sidebar on the right showing streaming video data and options to break execution.

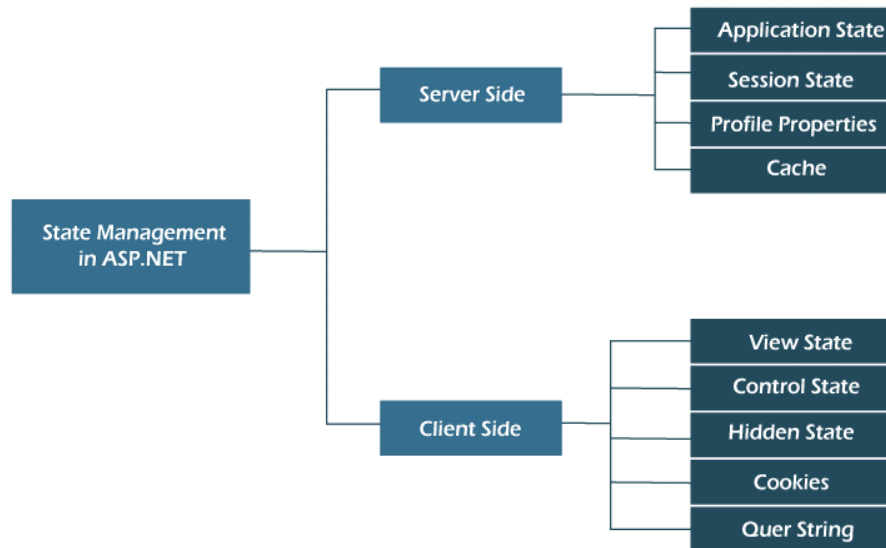
```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Chart ID="Chart1" runat="server" DataSourceID="SqlDataSource1">
<series>
<asp:Series Name="Series1" XValueMember="sname" YValueMembers="sid">
</asp:Series>
</series>
<chartareas>
<asp:ChartArea Name="ChartArea1">
</asp:ChartArea>
</chartareas>
</asp:Chart>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ConnectionString %%" SelectCommand="SELECT * FROM [student]"></asp:SqlDataSource>
</div>
</form>
</body>
</html>
    
```

Student Name	Value (sid)
Ankit	1
Deep	2
Sagor	3
Smit	4
Jeet	6

State Management in ASP.NET

- ✓ State Management is a process by which state and page information is maintained over multiple requests for same or different pages.
- ✓ As HTTP is a stateless protocol, server does not store any information once the response is sent back to client based on his request. When user submits request again, the server treats it as a new user. This is called stateless model.
- ✓ This model was workable when static web sites were developed and hosted in the past.
- ✓ Now, with interactive web sites or dynamic web site, there is a need to preserve some information to identify user, interact with user again and again within same session and same application. This concept is known as Statefull protocol. The information can be related to user, data objects, web pages or server objects.



State Management in ASP.NET

Server Side State Management Options

- ✓ **Application State:** Application state allows saving of data at application level which is accessible throughout the life of an application. The life of application starts when IIS is started and ends when IIS is stopped.
- ✓ **Session State:** The session state persists till the user is active or session time expires. When a user submits request, a session is started. The session gets over either when time expires or user explicitly abandons the sessions. The required information can be saved in session for later user by same user within the same application.
- ✓ **Profile Properties:** This option also allows saving of user specific data. It is similar to Session state except the data stored in Profile state never expires on use this property, the SQLProfileProvider class needs to be configured. It allows storing of data in SQL database. Since data is stored in database and not in application memory, therefore there is no risk of losing data even if IIS is restarted again and again.
- ✓ **Cache:** Caching is a technique by which frequently used data and web pages are stored in cache so that repeated cost of retrieval can be avoided. Storing frequently used data in cache ensures high availability, improved performance and scalability. Cache is object of System.Web.Caching Cache class. The main disadvantage of using Cache is that it is unreliable. The previously stored data stored in cache is deleted automatically to meet memory requirement of current process.

State Management in ASP.NET

Client Side State Management Options

- ✓ **View State:** View state provides facility to preserve page and values of controls at the client side. The information is stored after post back. Post back is a request from user for the page which is not for the first time. If value of IsPostBack property is true, it means page is not requested for the first time. The view state can be at page level, application level, machine level and control level. In page level state management, as long as the user is on current page, the information is retained. Whenever user submits form, the current state of page and controls are hashed into a string and saved in hidden field of the page. More than one hidden field can be used if data exceed limit set by MaxPageStateFieldLength property. When page is sent to server, the page parses the view state string and restores the information. This is default mechanism. The view state can be disabled at any stage. The property EnableViewState="false" is used in code when it is required to disable view state.
- ✓ **Control State:** This is another client side state management option. This is used when there is a need to store control data related to Custom control. View state can be disabled but control state cannot be disabled.
- ✓ **Hidden Field State:** ASP.NET allows storing information in a hidden field which is a server control and can be used to store information at page level. The value of the hidden field is sent to HTTP form collection along with value of other controls. The hidden file can be created in source file as given below. `<input type="hidden" id="username" name="username" value="">`
- ✓ **Cookies:** Cookie is a small amount of information that is stored in client machine.
- ✓ **QueryString:** A QueryString contains the information that is sent to server with URL.

State Management in ASP.NET

Client Side State Management Options

- ✓ **View State:** View state provides facility to preserve page and values of controls at the client side. The information is stored after post back. Post back is a request from user for the page which is not for the first time. If value of IsPostBack property is true, it means page is not requested for the first time. The view state can be at page level, application level, machine level and control level. In page level state management, as long as the user is on current page, the information is retained. Whenever user submits form, the current state of page and controls are hashed into a string and saved in hidden field of the page. More than one hidden field can be used if data exceed limit set by MaxPageStateFieldLength property. When page is sent to server, the page parses the view state string and restores the information. This is default mechanism. The view state can be disabled at any stage. The property EnableViewState="false" is used in code when it is required to disable view state.
- ✓ **Control State:** This is another client side state management option. This is used when there is a need to store control data related to Custom control. View state can be disabled but control state cannot be disabled.
- ✓ **Hidden Field State:** ASP.NET allows storing information in a hidden field which is a server control and can be used to store information at page level. The value of the hidden field is sent to HTTP form collection along with value of other controls. The hidden file can be created in source file as given below. `<input type="hidden" id="username" name="username" value="">`
- ✓ **Cookies:** Cookie is a small amount of information that is stored in client machine.
- ✓ **QueryString:** A QueryString contains the information that is sent to server with URL.

View State in ASP.NET

- ✓ View State is one of the methods of the ASP.NET page framework used to preserve and store the page and control values between round trips. It is maintained internally as a hidden field in the form of an encrypted value and a key.
- ✓ Default enables the View State for a page.
- ✓ The View State methods differ from the cache and cookies because the cookies are accessible from all the pages on your website, while the View State values are non-transferable, and thus you cannot access from different pages.
- ✓ Retain the control value on a page without storing them in a user profile or session state.
- ✓ Store page values and control properties that you set or define on a page.
- ✓ Create a custom View State Provider to store the view page information in a SQL Server Database or another database.
- ✓ View State can handle several data objects. Like String, Boolean Value, Array Object, Hash Table, and Custom type Converters

Advantages of View State

- ✓ Easy to Implement.
- ✓ No server resources are required: The View State is contained in a structure within the page load.
- ✓ Enhanced security features: It can be encoded and compressed or Unicode implementation.

Disadvantages of View State

- ✓ Security Risk: The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding. If security is a concern then consider using a Server-Based state Mechanism so that no sensitive information is sent to the client.
- ✓ Performance: Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- ✓ Device limitation: Mobile Devices might not have the memory capacity to store a large amount of View State data.

View State in ASP.NET

- ✓ View State is one of the methods of the ASP.NET page framework used to preserve and store the page and control values between round trips. It is maintained internally as a hidden field in the form of an encrypted value and a key.
- ✓ Default enables the View State for a page.
- ✓ The View State methods differ from the cache and cookies because the cookies are accessible from all the pages on your website, while the View State values are non-transferable, and thus you cannot access from different pages.
- ✓ Retain the control value on a page without storing them in a user profile or session state.
- ✓ Store page values and control properties that you set or define on a page.
- ✓ Create a custom View State Provider to store the view page information in a SQL Server Database or another database.
- ✓ View State can handle several data objects. Like String, Boolean Value, Array Object, Hash Table, and Custom type Converters

Advantages of View State

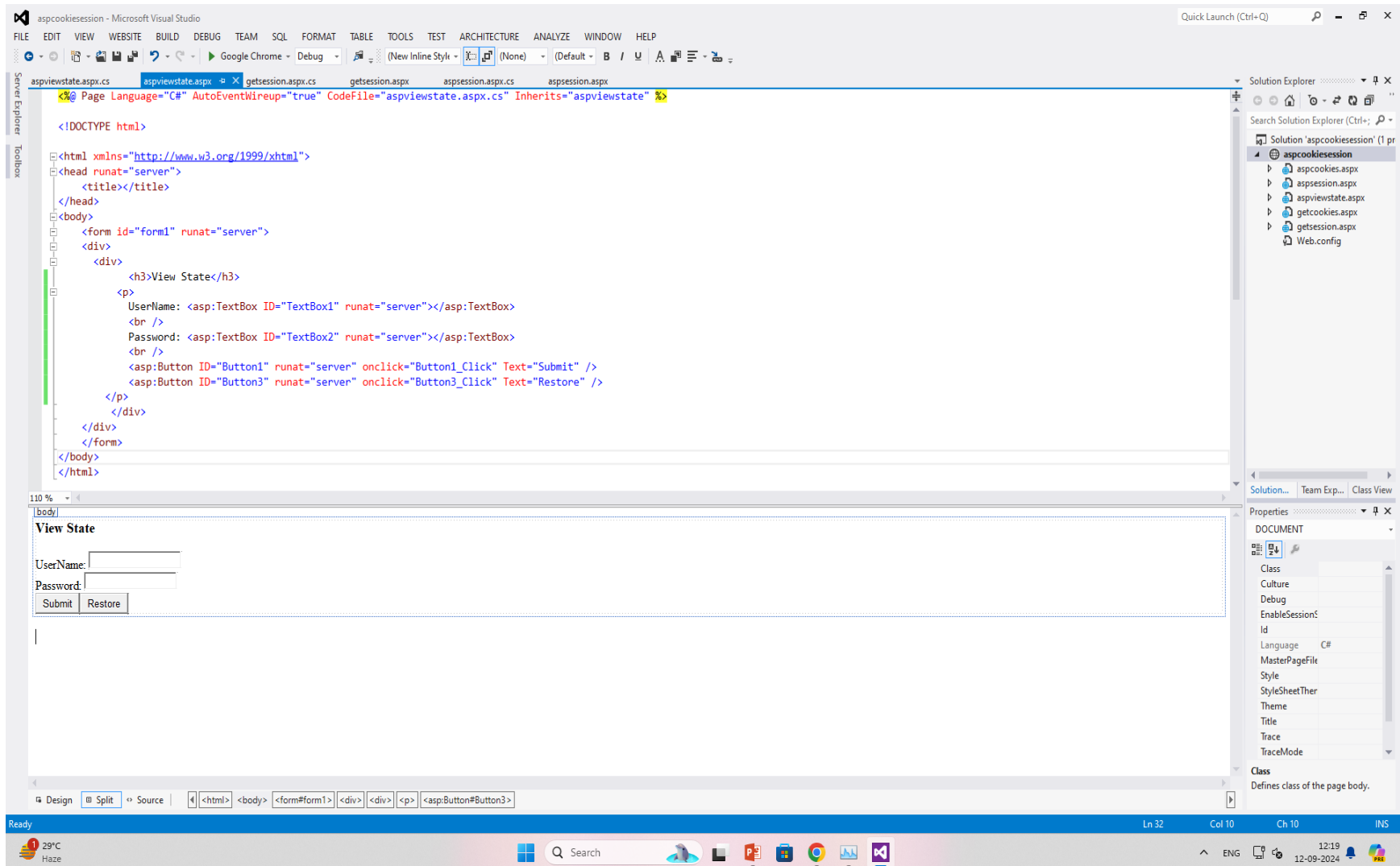
- ✓ Easy to Implement.
- ✓ No server resources are required: The View State is contained in a structure within the page load.
- ✓ Enhanced security features: It can be encoded and compressed or Unicode implementation.

Disadvantages of View State

- ✓ Security Risk: The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding. If security is a concern then consider using a Server-Based state Mechanism so that no sensitive information is sent to the client.
- ✓ Performance: Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- ✓ Device limitation: Mobile Devices might not have the memory capacity to store a large amount of View State data.

📢 View State Example in ASP.NET

Design View



The screenshot displays the Microsoft Visual Studio IDE in Design View for an ASP.NET page. The main window shows the rendered form with the following elements:

- Title:** View State
- Form:** A form with a border containing:
 - UserName:
 - Password:
 - Submit
 - Restore

The Solution Explorer on the right shows the project structure for 'aspcookieession' (1 project):

- aspcookies.aspx
- aspcookies.serveur
- aspviewstate.aspx
- getcookies.aspx
- getsession.aspx
- Web.config

The Properties window on the right shows the 'DOCUMENT' properties for the selected element:

Property	Value
Class	
Culture	
Debug	
EnableSessionState	
Id	
Language	C#
MasterPageFile	
Style	
StyleSheetTher	
Theme	
Title	
Trace	
TraceMode	

The status bar at the bottom shows 'Ready', '29°C Haze', and system information: 'Ln 32 Col 10 Ch 10 INS'.

📢 View State Example in ASP.NET

Code View & Output

Code View:

```

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

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

    protected void Button1_Click(object sender, EventArgs e)
    {
        ViewState["name"] = TextBox1.Text;
        ViewState["password"] = TextBox2.Text;
        // After clicking on Button, TextBox value will be cleared
        TextBox1.Text = "";
        TextBox2.Text = "";
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        if (ViewState["name"] != null)
        {
            TextBox1.Text = ViewState["name"].ToString();
        }

        if (ViewState["password"] != null)
        {
            TextBox2.Text = ViewState["password"].ToString();
        }
    }
}

```

Output:

View State

UserName:

Password:

Click on Submit Button Value Store in View State and Text Box Clear

View State

UserName:

Password:

Click on Restore Button Value Automatically Retrieve form View State to Text Box

Cookies in ASP.NET

- ✓ Cookies is a small piece of information stored on the client machine.
- ✓ This file is located on client machines "C:\Document and Settings\Currently_Login user\Cookie" path.
- ✓ It is used to store user preference information like Username, Password, City, PhoneNo, etc, on client machines.
- ✓ We need to import a namespace called System.Web.HttpCookie before we use cookie.

Type of Cookies

- ✓ **Persist Cookie** - A cookie that doesn't have expired time is called a Persist Cookie.
- ✓ **Non-Persist Cookie** - A cookie which has expired time is called a Non-Persist Cookie.

Cookie's common property

- ✓ Domain = This is used to associate cookies to domain.
- ✓ Secure = We can enable secure cookie to set true(HTTPs).
- ✓ Value = We can manipulate individual cookie.
- ✓ Values = We can manipulate cookies with key/value pair.
- ✓ Expires = This is used to set expire date for the cookies.

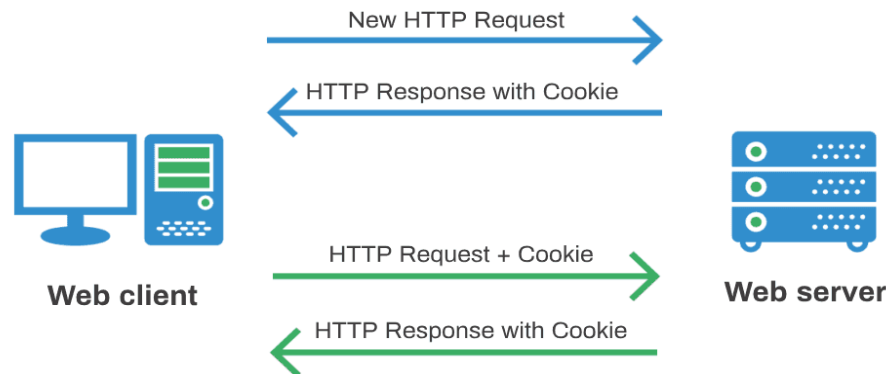
Advantages of Cookie

- ✓ It has clear text so the user can read it.
- ✓ We can store user preference information on the client machine.
- ✓ It is an easy way to maintain.
- ✓ Fast accessing.

Disadvantages of Cookie

- ✓ If the user clears the cookie information, we can't get it back.
- ✓ No security.
- ✓ Each request will have cookie information with page.

Cookies in ASP.NET



Create Cookie

```
HttpCookie c1 = new HttpCookie("c1");
```

Add Values in Cookies

```
c1["UserName"] = "Ankit Rami";
```

```
Response.Cookies["userName"].Value = "Ankit Rami";
```

Retrieve Cookie Value

```
Method-1 string User_Name = string.Empty;
```

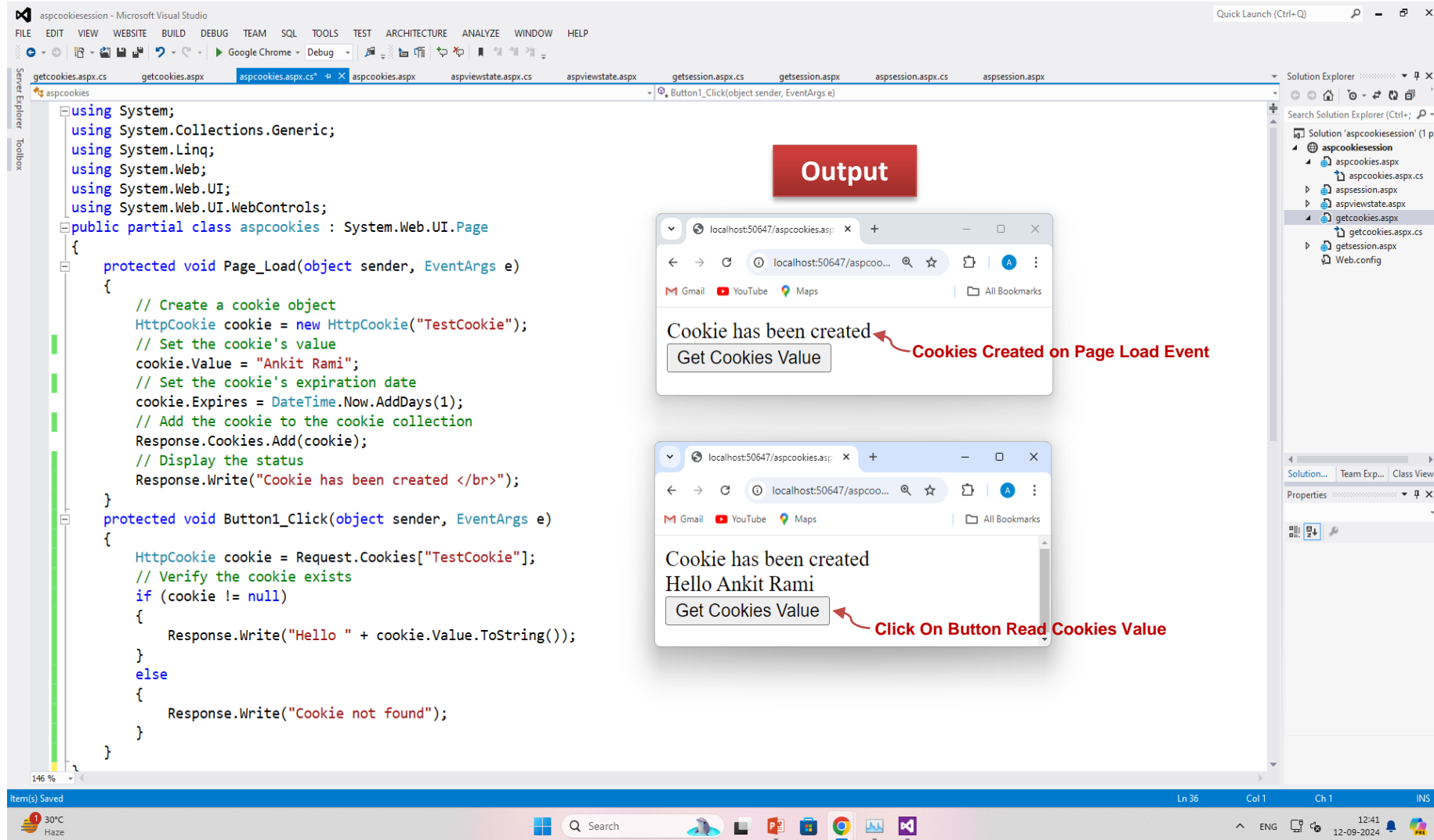
```
Method-2 User_Name = Request.Cookies["userName"].Value;
```

Clear the cookie

```
c1.Expires = DateTime.Now.AddHours(-1);
```

Cookies Example in ASP.NET

Code View



The screenshot displays the Visual Studio IDE with the following code in `aspcookies.aspx.cs`:

```

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

public partial class aspcookies : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Create a cookie object
        HttpCookie cookie = new HttpCookie("TestCookie");
        // Set the cookie's value
        cookie.Value = "Ankit Rami";
        // Set the cookie's expiration date
        cookie.Expires = DateTime.Now.AddDays(1);
        // Add the cookie to the cookie collection
        Response.Cookies.Add(cookie);
        // Display the status
        Response.Write("Cookie has been created <br>");
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        HttpCookie cookie = Request.Cookies["TestCookie"];
        // Verify the cookie exists
        if (cookie != null)
        {
            Response.Write("Hello " + cookie.Value.ToString());
        }
        else
        {
            Response.Write("Cookie not found");
        }
    }
}

```

Output

The browser output shows the following sequence of events:

- Page Load Event:** The browser displays "Cookie has been created" and a "Get Cookies Value" button. A red arrow points to this output with the text "Cookies Created on Page Load Event".
- Click On Button:** After clicking the "Get Cookies Value" button, the browser displays "Cookie has been created" followed by "Hello Ankit Rami" and the "Get Cookies Value" button. A red arrow points to this output with the text "Click On Button Read Cookies Value".

Session in ASP.NET

- ✓ ASP.NET session is a state that is used to store and retrieve values of a user.
- ✓ It helps to identify requests from the same browser during a time period (session).
- ✓ It is used to store value for the particular time session. By default, ASP.NET session state is enabled for all ASP.NET applications.
- ✓ Each created session is stored in SessionStateItemCollection object. We can get current session value by using Session property of Page object.
- ✓ A Session can store the value on the Server.
- ✓ It can support any type of object to be stored along with our own custom objects.
- ✓ A session is one of the best techniques for State Management because it stores the data as client-based.

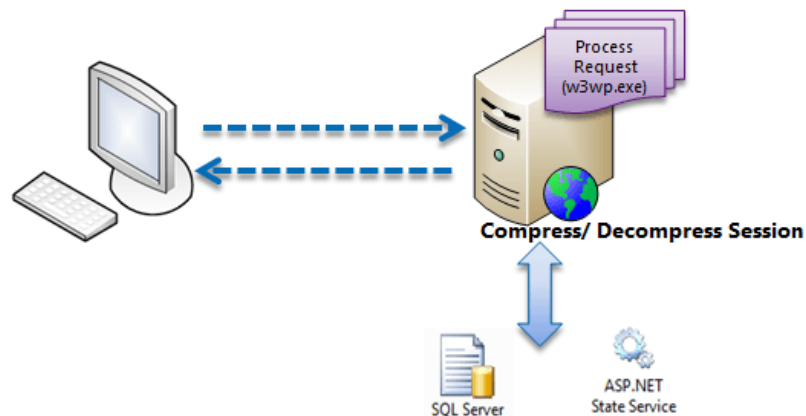
Advantages of Session

- ✓ It is easy to implement.
- ✓ Store data separately.
- ✓ Session is secure and transparent from the user.

Disadvantages of Session

- ✓ Performance decrease if we do not use InProc Session mode.
- ✓ Overhead involved in serializing and de-serializing session data.

📢 Session in ASP.NET



Create Session

```
Session["UserName"] = "Ankit Rami";
```

Retrieve Session Value

```
Response.Write(Session["UserName"].ToString());
```

Remove Session

Session.Remove(strSessionName):- Removes an item from the session state collection

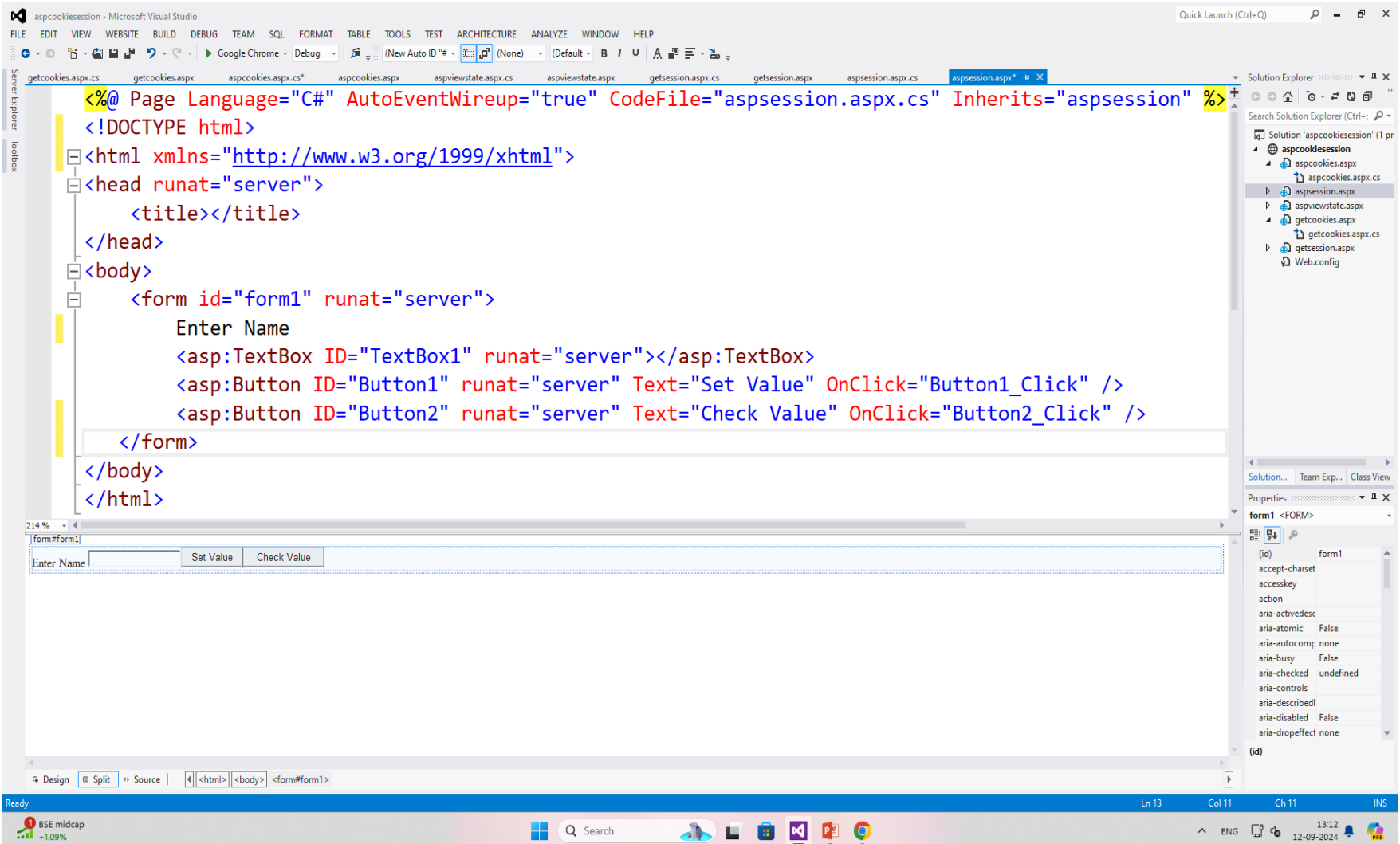
Session.RemoveAll():- Removes all items from the session collection.

Session.Clear():- it is same as session.RemoveAll() method.

Session.Abandon():- Cancels the current session.

📢 Session Example in ASP.NET

Set Session Design View



The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the source code for `aspession.aspx`. The code includes a page directive, DOCTYPE, HTML structure, and a form with a text box and two buttons.


```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="aspession.aspx.cs" Inherits="aspession" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
  Enter Name
  <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
  <asp:Button ID="Button1" runat="server" Text="Set Value" OnClick="Button1_Click" />
  <asp:Button ID="Button2" runat="server" Text="Check Value" OnClick="Button2_Click" />
</form>
</body>
</html>
```
- Design View:** Shows a visual representation of the form, including the text "Enter Name", a text box, and two buttons labeled "Set Value" and "Check Value".
- Solution Explorer:** Shows the project structure for "aspcookiesession", including files like `aspcookies.aspx`, `aspviewstate.aspx`, `getcookies.aspx`, `getsession.aspx`, and `Web.config`.
- Properties Window:** Shows the properties for the selected `form1` control, such as `accept-charset`, `accesskey`, `action`, and `aria-activedesc`.

📢 Session Example in ASP.NET

Set Session Code View

The screenshot displays the Visual Studio IDE with the following code in the `aspession.aspx.cs` file:

```

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

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

    protected void Button1_Click(object sender, EventArgs e)
    {
        Session["UserName"] = TextBox1.Text;
        Response.Write("Session Created");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        Response.Redirect("getsession.aspx");
    }
}

```

The browser output window shows the following content:

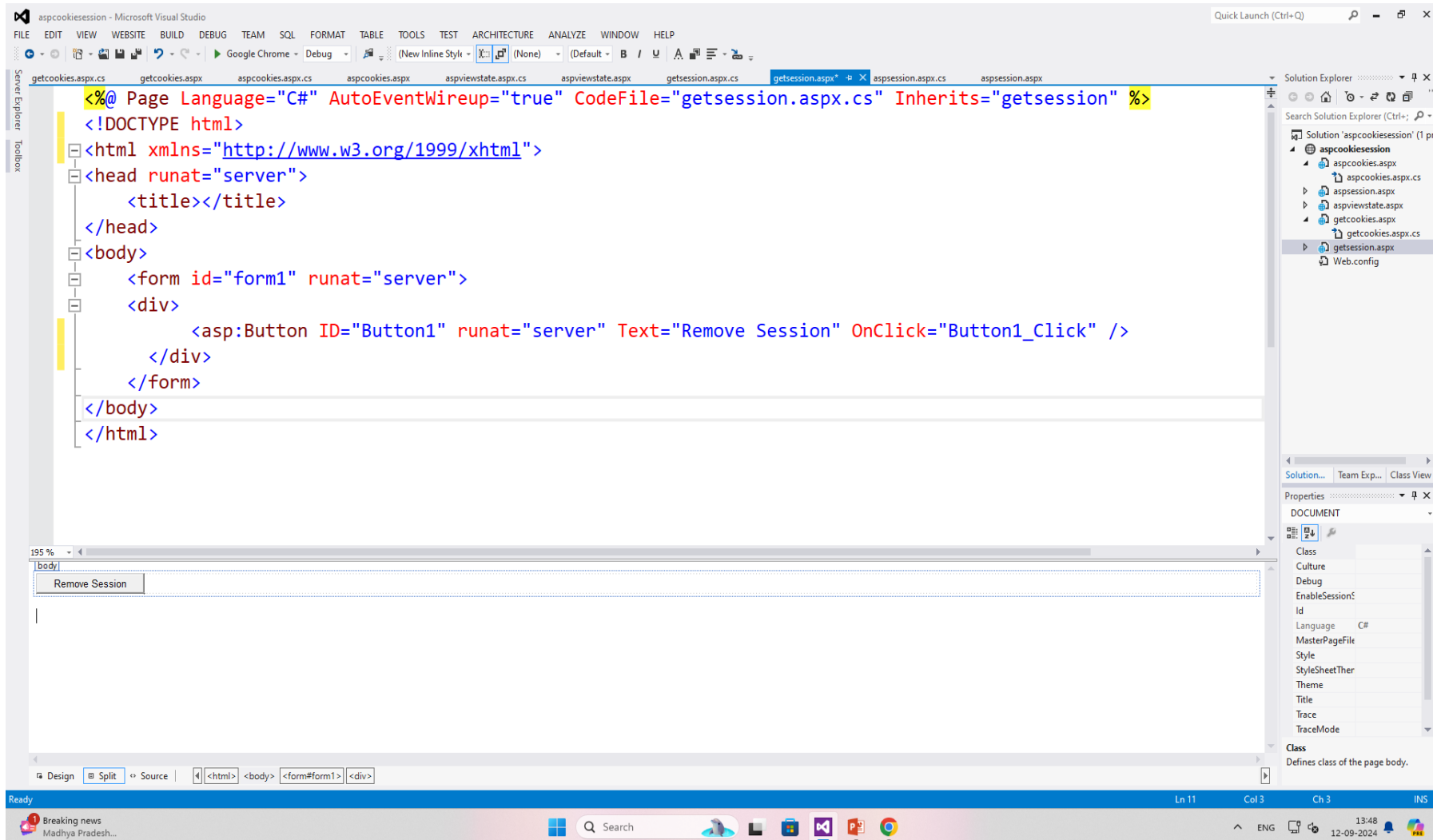
Output

Session Created ← Session Created on Button Click Event

Enter Name

Session Example in ASP.NET

Get Session Design View



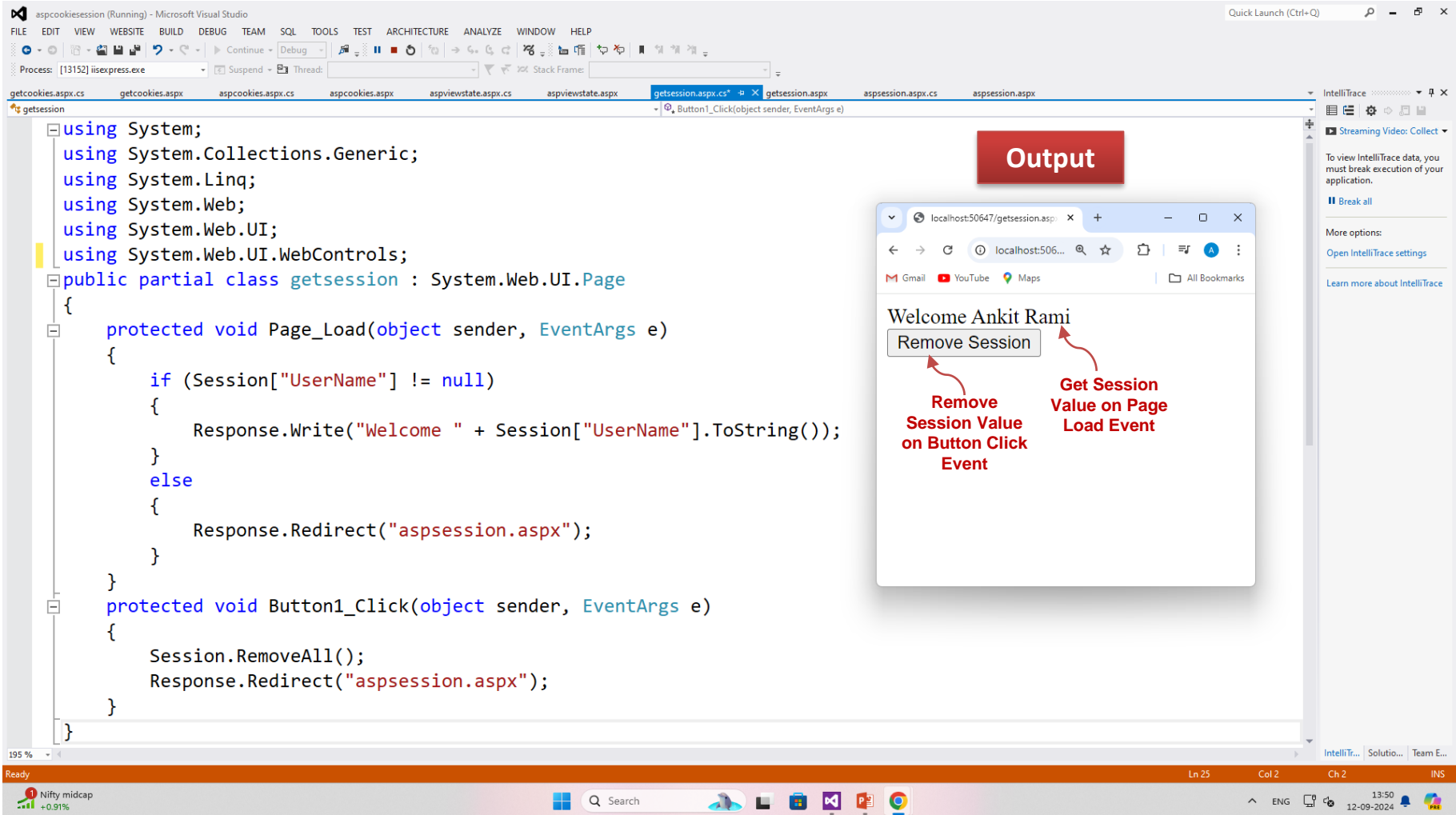
The screenshot displays the Microsoft Visual Studio IDE with the 'Design' view of an ASP.NET page. The page contains a single button labeled "Remove Session". The code-behind file is "getsession.aspx.cs". The page is titled "aspcookiesession" and is located in the "aspcookiesession" project.

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="getsession.aspx.cs" Inherits="getsession" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Button ID="Button1" runat="server" Text="Remove Session" OnClick="Button1_Click" />
</div>
</form>
</body>
</html>
  
```

Session Example in ASP.NET

Get Session Code View



The screenshot displays the Visual Studio IDE with a code editor on the left and a browser window on the right. The code editor shows the following C# code for a partial class named `getsession`:

```

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

public partial class getsession : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["UserName"] != null)
        {
            Response.Write("Welcome " + Session["UserName"].ToString());
        }
        else
        {
            Response.Redirect("aspession.aspx");
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Session.RemoveAll();
        Response.Redirect("aspession.aspx");
    }
}

```

The browser window shows the output of the application. The page displays "Welcome Ankit Rami" and a button labeled "Remove Session". Red arrows point from the browser output to the corresponding code in the editor:

- An arrow points from the "Remove Session" button in the browser to the `Button1_Click` method in the code, with the text "Remove Session Value on Button Click Event".
- Another arrow points from the "Welcome Ankit Rami" text in the browser to the `Page_Load` method in the code, with the text "Get Session Value on Page Load Event".

A red box labeled "Output" is positioned above the browser window. The browser's address bar shows `localhost:50647/getsession.aspx`. The Visual Studio interface includes the menu bar, toolbar, and solution explorer.

Application State in ASP.NET

- ✓ The ASP.NET application is the collection of all web pages, code and other files within a single virtual directory on a web server. When information is stored in application state, it is available to all the users.
- ✓ To provide for the use of application state, ASP.NET creates an application state object for each application from the `HttpApplicationState` class and stores this object in server memory. This object is represented by class file `global.asax`.
- ✓ Application State is mostly used to store hit counters and other statistical data, global application data like tax rate, discount rate etc. and to keep the track of users visiting the site.
- ✓ Application state is a data repository available to all classes in an ASP.NET application. Application state is stored in memory on the server and is faster than storing and retrieving information in a database. Unlike session state, which is specific to a single user session, application state applies to all users and sessions.
- ✓ Application state is stored in an instance of the `HttpApplicationState` class. This class exposes a key-value dictionary of objects.
- ✓ Application state variables are also used to store data when navigating from one page to another. It's multi-user Global data meaning it will be accessible across all pages and all sessions.
- ✓ Application state variables are stored on the web server in ASP.NET worker process memory.
- ✓ **Web farm:** It's a situation where the web application is deployed on different servers with load balancer.
- ✓ **Web garden:** It's a situation where the web application is deployed on the same server with multiple worker processes.

Advantages of application state:

- ✓ Application variable data is multi-user global data stored in memory.
- ✓ Easy to access.
- ✓ Fast retrieval.

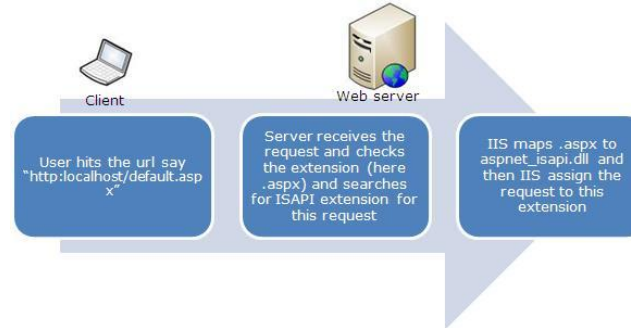
Disadvantages of application state:

- ✓ Application variable data is not able to survive the IIS restart and worker process recycling.
- ✓ Not suited for web farm and web garden like deployment situation.

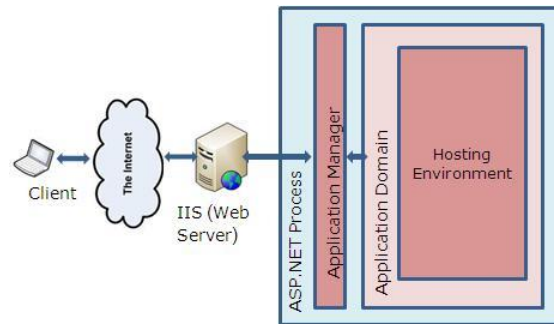
Application State in ASP.NET

Application State Life Cycle

Step 1: When the Browser sends a request to the web server and the server receives the request it first checks the extension to determine whether or not it is ISAPI because this request can only be handled by the ISAPI extension; if the extension is different then the request is handled by the server itself.



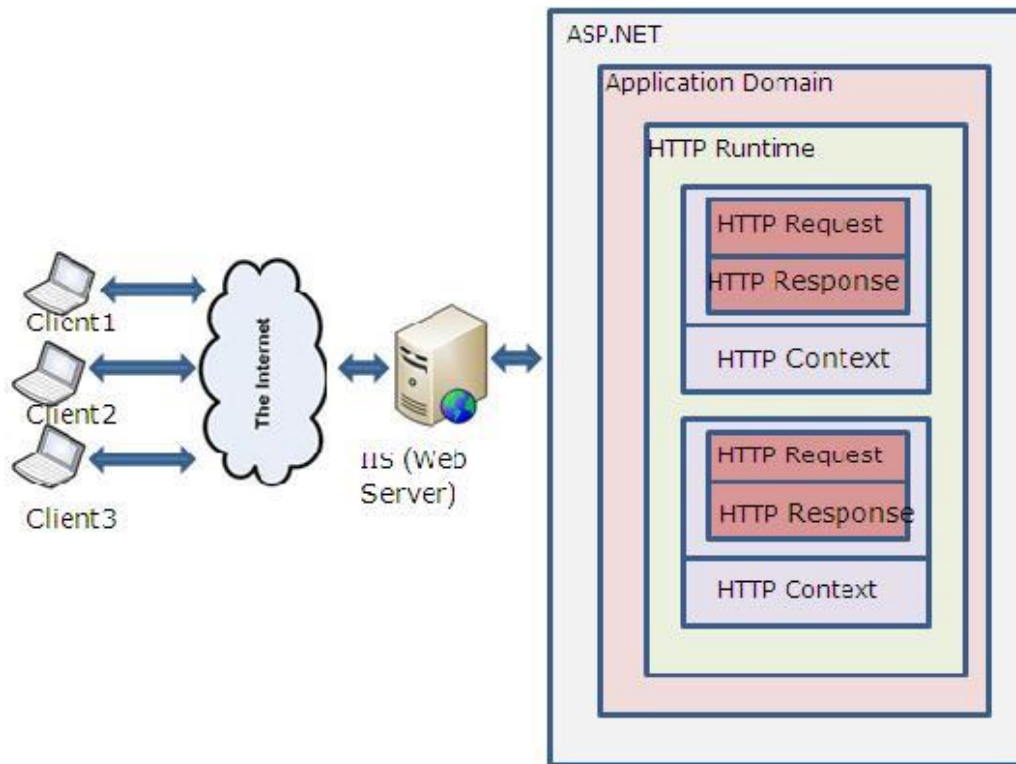
Step 2: After receiving the request the Application Manager creates an application domain. In the application domain an instance of the class HostingEnvironment is created that provides access to information about all application resources.




Application State in ASP.NET

Step 3: After creating the application domain, ASP.NET initializes the basic objects as HttpContext, HttpRequest and HttpResponse. HttpContext holds objects to the specific application request as HttpRequest and HttpResponse. HttpRequest contains all the information regarding the current request like cookies, browser information and so on and the HttpResponse contains the response that is sent to the client.

Step 4: Here all the basic objects are being initialized and the application is being started with the creation of the HTTPApplication class.



Application State in ASP.NET

Step 5: Then events are executed by the HTTPApplication class for any specific requirement. Here is a list of events: 

Global.asax file: the Global.asax file is used for handling application events or methods. It always exists in the root level. Events are one of the following of the 2 types in the Global application:

Events that will be raised on a certain condition.

Events that will be raised on every request.

The application will be started only once; if 10 users send a request then 10 user sessions are created. The events of the Global.asax file are:

Application_Start() : This method is invoked initially when first application domain is created.

Session_Start() : This method is called every time a session is start.

Application_BeginRequest() : After an application has started the first method Application_BeginRequest() is executed for every user.

Application_AuthenticateRequest() : It checks to determine whether or not the user is valid.

Application_Error() : Whenever an unhandled exception occurs then this event will be called.

Session_End() : When a user session is ended and all the data related to a specific user is cleared then the Session_End() event is called.

Application_End() : This method is called before the application ends. This can take place if IIS is restarted or the application domain is changing.

Application_Disposed() : This event is called after the application will be shut down and the .NET GC is about to reclaim the memory it occupies.

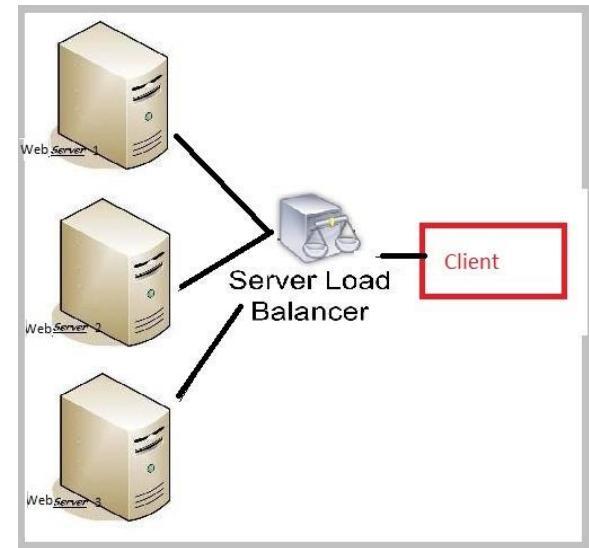
Although this is very late to perform any clean-up but we can use it for safety purposes.



📢 Application State in ASP.NET

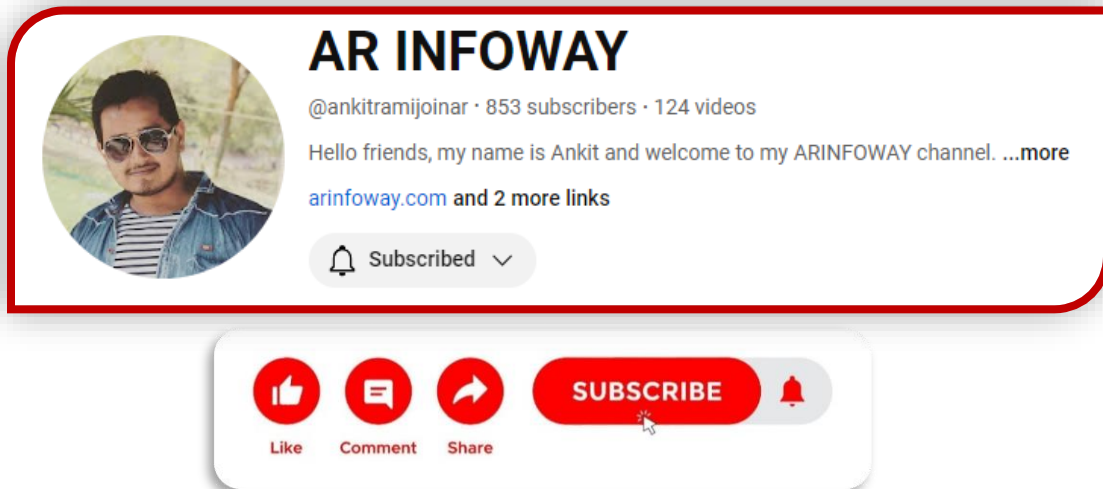
Important points of Application State variables

- ✓ Application State variables are available across all pages and all sessions. Application State variables are like multi-user Global data.
- ✓ Application variables are stored on a web server.
- ✓ Application State variables are cleared, only when the process hosting the application is restarted, that is when the application is ended.
- ✓ Application State variables do not support web farms and web gardens: Application State variables are not supported by web farms.
- ✓ A client sends a request and the request goes to the load balancer and the load balancer sends a request to web server1 and the Application State variables are stored in a web server1. If the subsequent request is sent by the client again and the load balancer sends a request to web server2 and the Application State variables are not stored in web server2 then something. Web servers do not share application state variables.
- ✓ Application State variables have a concurrency problem so we need to synchronize the method by using the lock and unlock methods. So multiple thread problems are resolved since only one thread can do the work.
- ✓ An application variable is used only when the variable needs to have global access and when you need them for the entire time, during the lifetime of an application.



Reference Link for E-Learning

- ❖ <https://www.javatpoint.com/ado-net-introduction>
- ❖ <https://www.c-sharpcorner.com/article/data-source-controls-in-asp-net/>
- ❖ <https://www.c-sharpcorner.com/UploadFile/af3b72/different-data-bound-controls-used-in-Asp-Net/>
- ❖ <https://www.javatpoint.com/state-management-in-asp-net>
- ❖ https://www.tutorialspoint.com/asp.net/asp.net_managing_state.htm
- ❖ <https://www.c-sharpcorner.com/UploadFile/225740/application-state-in-Asp-Net/>
- ❖ <https://amit.arinfoaway.com>
- ❖ <https://www.youtube.com/channel/UCWbJh2iQ8w-8nrU0Xpjpw7g>



AR INFOWAY
@ankitramijoinar · 853 subscribers · 124 videos
Hello friends, my name is Ankit and welcome to my ARINFOWAY channel. ...more
[arinfoaway.com](https://amit.arinfoaway.com) and 2 more links

Subscribed

Like Comment Share

SUBSCRIBE



AMIT ACADEMY
for Computer Education

**More Information Join
YouTube Channel**

View More