**505 .NET TECHNOLOGY**

**Assignment – 1**

**Q1) Explain Validation Controls.**

**Answer:**  
Validation controls in ASP.NET are used to ensure that the data entered by users in web forms is valid before it is submitted to the server. They reduce errors and improve data accuracy.

**Types of Validation Controls:**

1. **RequiredFieldValidator** – Ensures a field is not left empty.
   * Example: A user must enter an email before submission.
2. **RangeValidator** – Ensures input value is within a specific range.
   * Example: Age must be between 18 and 60.
3. **CompareValidator** – Compares input value with another control or constant.
   * Example: Password and Confirm Password must match.
4. **RegularExpressionValidator** – Validates input against a pattern.
   * Example: Check email format (abc@xyz.com).
5. **CustomValidator** – Allows custom validation logic using server or client-side code.
6. **ValidationSummary** – Displays a summary of all validation errors in one place.

➡️ These controls improve data reliability and reduce the chance of invalid inputs.

**Q2) Explain Session and State Management.**

**Answer:**  
**State Management** refers to preserving user data across multiple requests in a web application (since HTTP is a stateless protocol).

**Types:**

1. **Client-Side State Management:** Data is stored on the client side.
   * **ViewState**: Stores data in a hidden field on the page.
   * **Cookies**: Small files stored in the browser.
   * **Query Strings**: Data passed in the URL.
2. **Server-Side State Management:** Data is stored on the server.
   * **Session State**: Stores user-specific data (shopping cart, login info).
   * **Application State**: Stores data shared across all users.
   * **Cache**: Stores frequently used data temporarily for performance.

**Example (Session):**

Session["username"] = "Zack"; // Storing data

string user = Session["username"].ToString(); // Retrieving data

➡️ Session and State Management ensure smooth user experience across multiple pages.

**Assignment – 2**

**Q1) Explain Web.config.**

**Answer:**  
**Web.config** is an XML-based configuration file used in ASP.NET applications. It defines settings for the web application such as security, authentication, database connections, and custom error handling.

**Features of Web.config:**

* Contains **connection strings** for databases.
* Stores **session state and application settings**.
* Configures **authentication and authorization** rules.
* Handles **custom error messages**.
* Supports **caching configuration**.

**Example (Web.config snippet):**

<configuration>

<connectionStrings>

<add name="MyDB" connectionString="Data Source=.;Initial Catalog=StudentDB;Integrated Security=True"/>

</connectionStrings>

<system.web>

<authentication mode="Forms"/>

<customErrors mode="On"/>

</system.web>

</configuration>

➡️ Every ASP.NET application can have its own Web.config file for customization.

**Q2) Explain .NET Framework architecture.**

**Answer:**  
The **.NET Framework Architecture** is a software development platform developed by Microsoft. It provides a consistent environment for building and running applications.

**Main Components:**

1. **CLR (Common Language Runtime):**
   * Manages execution of code, memory, exceptions, garbage collection, and security.
2. **BCL (Base Class Library):**
   * Provides reusable classes (I/O, Collections, Data, Networking).
3. **Languages:**
   * Supports multiple languages (C#, VB.NET, F#).
4. **Application Domains:**
   * Isolates running applications for safety.
5. **Assemblies:**
   * Compiled code in DLL/EXE files.
6. **ASP.NET & ADO.NET:**
   * Frameworks for building web and data-driven applications.

➡️ The architecture ensures **language interoperability**, **security**, and **performance**.

**Assignment – 3**

**Q1) Explain Server Controls.**

**Answer:**  
Server Controls are special controls in ASP.NET that run on the server and render HTML on the client-side browser. They are more powerful than HTML controls because they support **events, state management, and data binding**.

**Types of Server Controls:**

1. **HTML Server Controls** – HTML elements with runat="server".
   * Example: <input type="text" runat="server" id="txtName"/>
2. **Web Server Controls** – Predefined ASP.NET controls.
   * Example: <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
   * Types: TextBox, Label, Button, CheckBox, DropDownList, GridView, etc.
3. **Validation Controls** – For data validation.
4. **Data Controls** – Like GridView, DataList, Repeater for database interaction.

➡️ Server controls simplify web development by managing events and maintaining state automatically.

**Q2) Explain ADO.NET architecture.**

**Answer:**  
**ADO.NET** (ActiveX Data Objects for .NET) is a data access technology in .NET used to interact with databases.

**Architecture Components:**

1. **Data Provider:** Connects to the database.
   * **Classes:** SqlConnection, SqlCommand, SqlDataAdapter, SqlDataReader.
2. **Connection Object:** Establishes a link to the database.
3. **Command Object:** Executes SQL queries or stored procedures.
4. **DataReader:** Provides fast, forward-only data retrieval.
5. **DataSet & DataTable:** Disconnected, in-memory storage of data.

**Example:**

SqlConnection con = new SqlConnection("connection\_string");

SqlCommand cmd = new SqlCommand("SELECT \* FROM Students", con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

while(dr.Read())

{

Console.WriteLine(dr["Name"].ToString());

}

con.Close();

➡️ ADO.NET is efficient for database connectivity, both online (connected) and offline (disconnected) operations.

**Assignment – 4**

**Q1) What is Postback? Explain example.**

**Answer:**  
**Postback** is the process in ASP.NET where the page sends data back to the server for processing and reloads the same page.

* **Happens when:** A user triggers an event (like clicking a button).
* **Example:** A form submission that reloads the same page with updated data.

**Example:**

<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit\_Click" />

protected void btnSubmit\_Click(object sender, EventArgs e)

{

lblMessage.Text = "Hello, " + txtName.Text;

}

➡️ When the button is clicked, the page posts back to the server and reloads with updated message.

**Q2) What is Compile Code? Explain Code Behind and Inline Coding.**

**Answer:**  
**Compile Code:**

* In .NET, source code is first compiled into **MSIL (Microsoft Intermediate Language)** by the compiler.
* At runtime, CLR converts MSIL into native machine code using **JIT (Just-In-Time Compiler)**.

**Code-Behind:**

* ASPX page (UI) is separated from the C# code file (.cs).
* Improves readability, reusability, and separation of design and logic.
* Example: Default.aspx + Default.aspx.cs.

**Inline Coding:**

* Code is written directly inside the ASPX page using <script runat="server">.
* Not recommended for large projects.
* Example:

<asp:Button ID="btn" runat="server" Text="Click" OnClick="btn\_Click" />

<script runat="server">

protected void btn\_Click(object sender, EventArgs e)

{

lbl.Text = "Inline Code Example";

}

</script>

➡️ **Difference:** Code-behind is preferred for large applications, while inline coding is useful for small projects or quick demos.