UNIT-II: Introduction to C# Programming with Respect to ASP.NET

1. Introduction to C# Programming with Respect to ASP.NET

1.1 Definition

C# (pronounced "C Sharp") is a modern, object-oriented programming language developed by Microsoft as part of its .NET framework. It is designed to be simple, secure, and robust, making it highly suitable for developing ASP.NET applications. ASP.NET is a web application framework provided by Microsoft that allows developers to build dynamic websites, web services, and web applications.

1.2 Exam Answer

What is C# Programming with respect to ASP.NET?
C# is a versatile and powerful programming language that works seamlessly within the .NET framework to create web applications using ASP.NET. It enables developers to write code that interacts with the ASP.NET runtime to manage web pages, handle user requests, and provide dynamic content generation. Together, C# and ASP.NET simplify the process of creating scalable, secure, and feature-rich web solutions.

1.3 Features of C# in ASP.NET Development

- **Object-Oriented**: Encourages modular, reusable code for ASP.NET applications.
- **Rich Library**: Provides an extensive set of libraries for managing data, UI, and web operations.
- **Type-Safe**: Reduces runtime errors by enforcing type checking during compile time.
- **Integrated with .NET**: Full compatibility with the .NET framework and CLR (Common Language Runtime).
- **Ease of Use**: Simplifies syntax and reduces development time.
- **Cross-Platform Development**: Supported via .NET Core for cross-platform ASP.NET applications.

1.4 Example

Creating a basic ASP.NET page using C#:

```
using System;
using System.Web.UI;

public partial class HelloWorld : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Hello, ASP.NET with C#!");
    }
}
```

2. Basics of ASP.NET

2.1 Definition

ASP.NET (Active Server Pages .NET) is a framework provided by Microsoft for developing web applications and services. Built on the .NET framework, it enables developers to build dynamic web pages that integrate easily with databases, APIs, and other services.

2.2 Exam Answer

What is ASP.NET? Explain its basics.
ASP.NET is a server-side web application framework designed to develop robust and dynamic websites, web applications, and services. It offers tools, libraries, and runtime support for building scalable web solutions. Developers can use ASP.NET to create data-driven applications that provide interactive and engaging user experiences.

2.3 Features of ASP.NET

1. **Cross-Platform Support**: Develop web apps for Windows, Linux, and macOS using .NET Core.
2. **Server-Side Scripting**: Processes requests and responses on the server for dynamic content generation.
3. **Rich Control Library**: Provides built-in server controls like GridView, DropDownList, etc.
4. **State Management**: Manages user data through view state, session state, and application state.
5. **Security**: Offers robust authentication and authorization mechanisms.
6. **Performance**: Highly optimized with features like Just-In-Time (JIT) compilation and caching.

2.4 Example

Creating a simple ASP.NET page:

```
<%@ Page Language="C#" %>
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to ASP.NET</title>
</head>
<body>
    <h1>Welcome to ASP.NET Basics</h1>
</body>
</html>
```

---

3. Creating and Deploying ASP.NET Applications

3.1 Definition

Creating and deploying ASP.NET applications involve designing the web application using tools like Visual Studio, coding using C# or VB.NET, and deploying the application to a web server like IIS (Internet Information Services).

3.2 Exam Answer

What is involved in creating and deploying ASP.NET applications?
Creating an ASP.NET application includes designing the application's user interface, writing server-side logic, and integrating with databases or APIs. Deployment is the process of publishing the application on a web server, making it accessible to users via the internet or an intranet.

3.3 Steps to Create an ASP.NET Application

1. **Set Up Development Environment**: Install Visual Studio and the required .NET SDK.
2. **Create a Project**: Choose a project type (e.g., ASP.NET Web Forms, MVC).
3. **Design the UI**: Use HTML, CSS, and ASP.NET controls for the user interface.
4. **Write Backend Logic**: Implement server-side functionality using C#.
5. **Test the Application**: Debug and test using built-in tools like IIS Express.

3.4 Steps to Deploy an ASP.NET Application

1. **Build the Project**: Generate the publish-ready files.
2. **Choose a Server**: Use IIS or cloud platforms like Azure.
3. **Publish the Application**: Upload the files to the server.
4. **Configure the Server**: Set up permissions and application settings.
5. **Access the Application**: Test by navigating to the URL.

---

4. Web Forms

4.1 Definition

ASP.NET Web Forms is a web development model that uses a drag-and-drop, event-driven programming approach for building dynamic web applications.

4.2 Exam Answer

What are Web Forms?
Web Forms is a part of the ASP.NET framework that enables developers to build web pages using a visual design interface and an event-driven programming model. It abstracts HTML, CSS, and JavaScript complexities, making it easier to create interactive web applications.

4.3 Features of Web Forms

- **Drag-and-Drop Interface**: Simplifies UI design.
- **Event-Driven Model**: Allows developers to handle user interactions using events.
- **State Management**: Maintains user data across requests.
- **Built-In Controls**: Provides a wide range of server controls like Buttons, Labels, and Grids.

4.4 Example

A simple Web Form with a Button:

```
<asp:Button ID="Button1" runat="server" Text="Click Me" OnClick="Button1_Click"
/>
```

---

5. Web Controls

5.1 Definition

Web controls in ASP.NET are server-side components that simplify the development of dynamic web pages. These controls abstract HTML elements and provide additional functionality.

5.2 Types of Web Controls

1. **Standard Controls**: E.g., Buttons, TextBoxes, Labels.
2. **Data Controls**: E.g., GridView, Repeater.
3. **Validation Controls**: E.g., RequiredFieldValidator, RangeValidator.

5.3 Example

Creating a TextBox with a Button:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1_Click"
/>
```

---

6. Working with Events

6.1 Definition

In ASP.NET, events are actions or occurrences, such as a button click or a page load, that can trigger the execution of specific server-side code. Events allow developers to create interactive and dynamic web applications by responding to user actions.

6.2 Exam Answer

What is "Working with Events" in ASP.NET?
Working with events in ASP.NET involves handling user-triggered actions, such as clicking a button or selecting an item from a dropdown, using event handlers. The ASP.NET framework provides a robust event-driven model that ensures efficient management of these interactions, allowing developers to execute server-side logic in response to user inputs.

6.3 Features of Event Handling in ASP.NET

- **Server-Side Execution**: Event handling logic is executed on the server.

- **Built-In Events**: A wide range of pre-defined events like `Click`, `Load`, `SelectedIndexChanged`, etc.
- **Customizable**: Developers can create custom events.
- **Event Bubbling**: Events propagate from child to parent controls.
- **Integration with Web Controls**: Most server controls come with built-in event support.

6.4 Common Events in ASP.NET

1. **Page Events**: Triggered during the lifecycle of a web page.
   - `Page_Load`: Executed when the page is loaded.
   - `Page_PreRender`: Executed just before the page is rendered to the client.
2. **Control Events**: Triggered by user interaction with server controls.
   - `Button.Click`: Occurs when a button is clicked.
   - `TextBox.TextChanged`: Occurs when the text in a TextBox changes.

6.5 Example: Handling a Button Click Event

ASPX Markup

```
<asp:Button ID="Button1" runat="server" Text="Click Me" OnClick="Button1_Click"
/>
```

Code-Behind (C#)

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("Button was clicked!");
}
```

When the button is clicked, the `Button1_Click` method executes, displaying a message on the page.

6.6 Advantages of Event Handling

- **Encapsulated Logic**: Keeps user interaction logic separate from presentation.
- **Reusability**: Event handlers can be reused across multiple controls.
- **Dynamic Interaction**: Enables the creation of interactive applications.

6.7 Disadvantages of Event Handling

- **Increased Server Load**: Each event triggers a server round trip.
- **Complexity in Debugging**: Debugging event-driven applications can be challenging due to multiple handlers.