CERTIFIED QUIZ

Introduction to ASP.NET

ASP.NET is an open-source, server-side web application framework developed by Microsoft. It is used to create dynamic web pages and web applications. ASP.NET allows developers to build web-based applications using .NET languages such as C# and VB.NET. ASP.NET is part of the .NET framework and provides a robust set of libraries and APIs for building scalable, secure, and maintainable web applications.

ASP.NET Architecture

ASP.NET follows a multi-tier architecture model, which separates the application into different layers, each responsible for different aspects of the application:

1. Presentation Layer: This layer handles the user interface (UI) and user interaction. It consists of web pages (e.g., ASPX files, Razor views).
2. Business Logic Layer (BLL): This layer contains the business logic and validation rules of the application.
3. Data Access Layer (DAL): This layer interacts with the database, handling CRUD operations.
4. Database: This is the layer where data is stored and retrieved from.

ASP.NET Web Forms

ASP.NET Web Forms is a development model for building web applications with a drag-and-drop, event-driven model. It allows developers to create dynamic web pages using familiar Windows-style controls (e.g., text boxes, buttons).

Key features:
- Code-behind model: The logic is separated from the HTML.
- Server-side controls: Provide rich UI elements that are processed on the server.
- ViewState: Maintains the state of web controls between postbacks.

Example:
aspx

ASP.NET MVC

ASP.NET MVC is a design pattern that separates an application into three main components:

1. Model: Represents the application's data and business logic.
2. View: Represents the UI that displays the data.
3. Controller: Handles user input and updates the Model and View accordingly.

ASP.NET MVC provides greater control over HTML, JavaScript, and CSS than Web Forms, making it ideal for developers looking for a lightweight framework.

Example:
csharp
```
public class HomeController : Controller {
public ActionResult Index() {
return View();
}
}
```

ASP.NET Core

ASP.NET Core is a modern, cross-platform, high-performance framework for building cloud-based and internet-connected applications. It is the successor to the original ASP.NET and is designed to work on Windows, macOS, and Linux.

Key features of ASP.NET Core include:
- Cross-platform: Runs on Windows, macOS, and Linux.
- High performance: Optimized for performance and scalability.
- Modular framework: Developers can add only the necessary libraries to reduce the app's size.

Example:
csharp
```
public class Startup {
public void Configure(IApplicationBuilder app) {
app.UseRouting();
app.UseEndpoints(endpoints => {
endpoints.MapGet("/", async context => {
await context.Response.WriteAsync("Hello World!");
});
});
}
}
```

Entity Framework

Entity Framework (EF) is an open-source ORM (Object-Relational Mapping) framework for .NET that simplifies data access by allowing developers to interact with databases using strongly-typed objects rather than writing raw SQL queries. It supports LINQ (Language-Integrated Query) for querying the database in a strongly-typed manner.

Example:
csharp
```
using (var context = new SchoolContext()) {
var students = context.Students.ToList();
}
```

Routing in ASP.NET Core

Routing in ASP.NET Core is the process of mapping incoming HTTP requests to the appropriate controller action. The routing system examines the URL and determines which controller and action should handle the request.

Example:
csharp
```
app.UseEndpoints(endpoints => {
endpoints.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Middleware

Middleware in ASP.NET Core is software that is executed on every HTTP request. It handles requests and responses in the application pipeline. Examples of middleware include authentication, logging, and error handling.

Example:

csharp

```
public void Configure(IApplicationBuilder app) {
app.Use(async (context, next) => {
await context.Response.WriteAsync("Request handled by middleware.");
await next.Invoke();
});
}
```