|  |  |
| --- | --- |
|  | **Trainer : Madan.Patakota****Topic : Asp.NetCore Interview Question and answers** ☎ : 7204185064  ✉ : [madan.patakota@gmail.com](mailto:madan.patakota@gmail.com) | [madan@misard.com](mailto:madan@misard.com)  **LinkedIn** : <https://www.linkedin.com/in/madanpatakota>  **GitHub** : <https://github.com/madanpatakota>  **You tube** : <https://www.youtube.com/@madan4422>  **Training** : <https://learn.cloudcamp.in/learn/Html>  **Websites** : https://misardcourses.azurewebsites.net  <https://misard.azurewebsites.net> |
|  |  |

**1. What is .NET Core?**

**Answer:** .NET Core is an open-source, cross-platform framework developed by Microsoft for building modern, cloud-based, and internet-connected applications. It supports multiple operating systems like Windows, Linux, and macOS.

**2. What is the difference between .NET Core and .NET Framework?**

**Answer:** .NET Core is cross-platform and open-source, while .NET Framework is only for Windows. .NET Core is suitable for cloud applications, while .NET Framework is used for desktop applications.

**3. Explain Middleware in .NET Core.**

**Answer:** Middleware is a software component that handles requests and responses in an application. In .NET Core, middleware components are assembled into an application pipeline to handle HTTP requests.

**4. What is Dependency Injection in .NET Core?**

**Answer:** Dependency Injection (DI) is a design pattern used to manage dependencies between objects. In .NET Core, DI is built-in and helps to create flexible, maintainable, and testable applications.

**5. What is the use of the Startup class in .NET Core?**

**Answer:** The Startup class configures services and the app's request pipeline. It includes methods like ConfigureServices and Configure to set up dependencies and middleware.

**6. How do you handle configuration in .NET Core?**

**Answer:** Configuration in .NET Core is handled using various sources like JSON files (appsettings.json), environment variables, and command-line arguments. The ConfigurationBuilder class is used to read these configurations.

**7. What are Razor Pages in ASP.NET Core?**

**Answer:** Razor Pages is a page-based programming model for building web UI in ASP.NET Core. It is simpler than MVC and is used for building web pages with server-side logic.

**8. What is Kestrel in .NET Core?**

**Answer:** Kestrel is a cross-platform web server for ASP.NET Core. It is lightweight, fast, and used as the default web server in .NET Core applications.

**9. How do you manage versions in .NET Core?**

**Answer:** Versions in .NET Core are managed using the global.json file, which specifies the .NET SDK version, and NuGet packages, which manage library versions.

**10. What is Entity Framework Core?**

**Answer:** Entity Framework Core (EF Core) is an ORM (Object-Relational Mapper) for .NET Core. It allows developers to work with a database using .NET objects, making data access simpler and more abstract.

**11. What are the main features of .NET Core?**

**Answer:** Key features include cross-platform support, high performance, flexible deployment, a unified programming model, and support for microservices and containers.

Yeah , There are some important features are in .net Core . So you can say , those are Key features include cross-platform support , high performance and you can say , flexible deployment and its supports the microservices.

**12. How do you create a new .NET Core project?**

**Answer:** Use the dotnet new command followed by the project template, like dotnet new console for a console app or dotnet new mvc for an MVC web app.

**13. What is the difference between IServiceCollection and IServiceProvider?**

**Answer:** IServiceCollection is used to register services, while IServiceProvider is used to resolve services.

**Yeah coming to the IServiceCollection its used to register the services and Iserviceprovider which is usefor resolve the sevices.**

**14. Explain the concept of Logging in .NET Core.**

**Answer:** Logging in .NET Core is a built-in feature that helps to log information for debugging and monitoring. It supports various providers like Console, Debug, and third-party providers like Serilog.

**15. What is gRPC in .NET Core?**

**Answer:** gRPC is a high-performance, open-source framework for remote procedure calls (RPC). It allows you to define services using protocol buffers and communicate between client and server.

**16. How do you use static files in .NET Core?**

**Answer:** To use static files, add the UseStaticFiles middleware in the Configure method of the Startup class. This serves files from the wwwroot folder by default.

**17. What is the purpose of the launchSettings.json file?**

**Answer:** launchSettings.json contains settings for how the application is launched, including profiles for different environments and configurations for web servers and ports.

**18. What is the .NET Core CLI?**

**Answer:** The .NET Core Command-Line Interface (CLI) is a set of tools for building, running, and managing .NET Core applications from the command line. Key commands include dotnet build, dotnet run, and dotnet publish.

**19. Explain the use of appsettings.json in .NET Core.**

**Answer:** appsettings.json is a configuration file used to store application settings like connection strings, logging levels, and other custom configurations. It is loaded automatically by the configuration system.

**20. How do you enable CORS in .NET Core?**

**Answer:** Enable CORS by adding the AddCors method in ConfigureServices and using the UseCors middleware in the Configure method of the Startup class. You can define policies to allow specific origins, methods, and headers.

These additional questions provide a broader understanding of .NET Core concepts and practices, suitable for a comprehensive interview preparation.

**21. What is the difference between IApplicationBuilder and IWebHostBuilder?**

**Answer:** IApplicationBuilder is used to configure the application's request pipeline, while IWebHostBuilder is used to configure and create the web host which manages the server and application lifetime.

**22. What is the purpose of IHostingEnvironment?**

**Answer:** IHostingEnvironment provides information about the web hosting environment an application is running in, such as environment name, content root path, and web root path.

**23. How do you handle exceptions in .NET Core?**

**Answer:** Use middleware like UseExceptionHandler for global exception handling and UseDeveloperExceptionPage for detailed error information during development.

**24. What is Blazor in .NET Core?**

**Answer:** Blazor is a framework for building interactive web UIs using C# instead of JavaScript. It supports both server-side and client-side hosting models.

**25. How do you implement authentication and authorization in .NET Core?**

**Answer:** Implement authentication using middleware like AddAuthentication and UseAuthentication.

Use AddAuthorization to set up authorization policies and UseAuthorization to enforce them.

We can Implement authentication using middleware like AddAuthentication and UseAuthentication mecanisms

And Use AddAuthorization to set up the authorization policies and UseAuthorization to enforce the polices.

**26. Explain the concept of Health Checks in .NET Core.**

**Answer:** Health Checks are used to monitor the health of an application and its dependencies. They provide endpoints to report the status of the application and can be configured using AddHealthChecks and MapHealthChecks.

**27. What is the role of the Program.cs file in a .NET Core application?**

**Answer:** Program.cs contains the Main method, which is the entry point of the application. It creates and configures the web host, which runs the application.

**28. How do you use Entity Framework Core with .NET Core?**

**Answer:** To use EF Core, add the necessary packages, create a DbContext class to represent the database context, define entity classes, and configure the DbContext in Startup.cs using AddDbContext.

**29. What are Tag Helpers in ASP.NET Core?**

**Answer:** Tag Helpers are server-side components that help render HTML elements in Razor views. They provide a way to attach server-side behavior to existing HTML elements.

**30. What is the difference between IActionResult and ActionResult<T>?**

**Answer:** IActionResult is a general interface for action results, while ActionResult<T> is a type-specific result that allows for more explicit response types and improved type safety in controller actions.

**31. What is the purpose of ConfigureServices in the Startup class?**

**Answer:** ConfigureServices is used to register application services with the dependency injection container. It allows you to configure services like Entity Framework, Identity, and custom services.

**32. Explain the difference between Transient, Scoped, and Singleton lifetimes in .NET Core DI.**

**Answer:**

* Scoped: A new instance is created per request/connection.  
  Example: Use services.AddScoped<IService, ServiceImplementation>(); to register a scoped service.
* Singleton: A single instance is created and shared throughout the application's lifetime.  
  Example: Use services.AddSingleton<IService, ServiceImplementation>(); to register a singleton service.
* Transient: A new instance is created each time it's requested.  
  Example : Example: Use services.AddTransient<IService, ServiceImplementation>(); to register a transient service.

**33. How do you create a RESTful API in .NET Core?**

**Answer:** Create a new ASP.NET Core Web API project, define controller classes that inherit from ControllerBase, and use attributes like [HttpGet], [HttpPost], [HttpPut], and [HttpDelete] to define API endpoints.

**34. What are Filters in ASP.NET Core?**

**Answer:** Filters are used to run code before or after specific stages in the request pipeline. They include Authorization, Resource, Action, and Exception filters, and are useful for cross-cutting concerns like logging, authorization, and error handling.

**35. How do you implement versioning in .NET Core Web API?**

**Answer:** Implement API versioning by installing the Microsoft.AspNetCore.Mvc.Versioning package and configuring versioning in Startup.cs using AddApiVersioning. Use attributes like [ApiVersion] and [MapToApiVersion] to manage versions.

**36. What is the purpose of the UseEndpoints method in .NET Core?**

**Answer:** UseEndpoints is used to configure endpoint routing. It defines the endpoints for the application and maps them to request delegates, controllers, or Razor Pages.

**37. How do you use SignalR in .NET Core?**

**Answer:** Install the SignalR package, add SignalR services in ConfigureServices, configure SignalR hubs in Configure, and create hub classes that inherit from Hub to handle real-time communication.

**38. What is a Razor Component in Blazor?**

**Answer:** A Razor Component is a self-contained unit of UI in Blazor, defined using .razor files. It combines HTML and C# code to create reusable UI elements.

**39. How do you handle database migrations in Entity Framework Core?**

**Answer:** Use EF Core CLI commands like dotnet ef migrations add to create migrations and dotnet ef database update to apply migrations. Migrations manage schema changes in the database.

**40. Explain the role of Middleware in request processing in .NET Core.**

**Answer:** Middleware processes requests and responses in the pipeline. Each middleware component can handle, modify, or pass requests and responses to the next component. The order of middleware registration defines the request processing sequence.

**41. What is .NET Core Hosting?**

**Answer:** .NET Core Hosting refers to the infrastructure that manages the lifecycle of a .NET Core application. This includes the setup and configuration of the web server, like Kestrel, and the ASP.NET Core middleware pipeline.

**42. How do you perform model validation in .NET Core?**

**Answer:** Model validation in .NET Core is done using Data Annotations attributes in your model classes. Attributes like [Required], [StringLength], and [Range] are used to enforce validation rules. The ModelState.IsValid property is checked in the controller to ensure the model is valid.

**43. Explain the concept of View Components in ASP.NET Core.**

**Answer:** View Components are similar to partial views but are more powerful. They encapsulate rendering logic and are reusable across different views. A View Component consists of a class that inherits from ViewComponent and a corresponding view.

**44. What is Kestrel in .NET Core?**

**Answer:** Kestrel is a cross-platform web server for ASP.NET Core. It is lightweight, fast, and is the default web server used in ASP.NET Core applications. Kestrel can be used directly or in combination with a reverse proxy like IIS, Nginx, or Apache.

**45. How do you enable HTTPS in an ASP.NET Core application?**

**Answer:** Enable HTTPS by configuring the app to use the UseHttpsRedirection middleware in the Configure method of the Startup class. Additionally, configure your hosting environment to support HTTPS and provide the necessary certificates.

**46. What is Swagger, and how is it used in .NET Core?**

**Answer:** Swagger is an open-source tool for documenting APIs. In .NET Core, the Swashbuckle package integrates Swagger. It generates interactive API documentation, making it easier to understand and test API endpoints.

**47. Explain Dependency Injection (DI) in .NET Core.**

**Answer:** Dependency Injection (DI) in .NET Core is a built-in framework that provides a way to achieve Inversion of Control (IoC) by injecting dependencies into classes rather than hard-coding them. It improves code maintainability and testability.

**48. What are Policy-based Authorization and Role-based Authorization in .NET Core?**

**Answer:**

* **Policy-based Authorization:** Defines access policies based on specific conditions and requirements. Policies are created and applied to controllers or actions.
* **Role-based Authorization:** Uses roles to restrict access. Users are assigned roles, and access is granted based on these roles.

**49. How do you implement Global Exception Handling in .NET Core?**

**Answer:** Global Exception Handling is implemented using the UseExceptionHandler middleware. Configure a central error handling page or logic in the Configure method of the Startup class to catch and handle exceptions globally.

**50. What is the purpose of the WebHost.CreateDefaultBuilder method?**

**Answer:** WebHost.CreateDefaultBuilder configures a .NET Core web application with default settings, including setting up Kestrel as the web server, loading configuration from appsettings.json and environment variables, and enabling logging.

**51. What are Endpoints in ASP.NET Core?**

**Answer:** Endpoints are a new way to configure and route requests in ASP.NET Core. They centralize route information and allow middleware to handle route matching and endpoint execution. Defined in UseEndpoints in the Startup class.

**52. How do you enable Session State in ASP.NET Core?**

**Answer:** To enable session state, add the AddSession and AddDistributedMemoryCache services in ConfigureServices. Use the UseSession middleware in Configure and configure session options as needed.

**53. What is the IHostedService interface in .NET Core?**

**Answer:** IHostedService is an interface for implementing background tasks in .NET Core applications. Services implementing this interface are run in the background, such as hosted services or timed services.

**54. Explain the difference between Model Binding and Model Validation in ASP.NET Core.**

**Answer:**

* **Model Binding:** The process of creating .NET objects from HTTP request data (form data, query string, route data).
* **Model Validation:** Ensures the created model object meets defined validation rules, typically using Data Annotations.

**55. What is the difference between IActionResult and IResult in .NET Core?**

**Answer:** IActionResult is used in controllers for returning different types of responses, allowing for flexible return types. IResult is a newer, more lightweight interface introduced for minimal APIs, focusing on simplicity and performance.

**56. How do you handle static files in ASP.NET Core?**

**Answer:** Use the UseStaticFiles middleware in the Configure method of Startup to serve static files like HTML, CSS, JavaScript, and images from the wwwroot folder.

**57. What is the purpose of the ConfigureWebHostDefaults method?**

**Answer:** ConfigureWebHostDefaults is used in ASP.NET Core 3.0 and later to set up default configurations for a web host, simplifying the process of setting up Kestrel, logging, and other common configurations.

**58. How do you use gRPC in .NET Core?**

**Answer:** To use gRPC, install the Grpc.AspNetCore package, define service contracts using .proto files, generate server and client code, and configure gRPC services in Startup using AddGrpc and MapGrpcService.

**59. What is Minimal API in .NET Core?**

**Answer:** Minimal API is a simplified way of building HTTP APIs in .NET Core with less ceremony. It uses fewer files and dependencies, focusing on minimal configuration and setup, ideal for small microservices.

**60. How do you configure Cross-Origin Resource Sharing (CORS) in ASP.NET Core?**

**Answer:** Configure CORS by adding the AddCors service in ConfigureServices, defining CORS policies, and using the UseCors middleware in Configure. Policies specify allowed origins, headers, and methods.

**61. What is Dependency Injection (DI)?**

**Answer:** Dependency Injection (DI) is a design pattern used to achieve Inversion of Control (IoC) between classes and their dependencies. It allows objects to be injected rather than instantiated within a class, promoting loose coupling and easier testing.

**62. Explain the difference between ControllerBase and Controller in ASP.NET Core.**

**Answer:** ControllerBase is a base class for MVC controllers without view support, typically used for APIs. Controller inherits from ControllerBase and adds support for views, used in MVC applications with Razor views.

**63. How do you implement caching in ASP.NET Core?**

**Answer:** Implement caching using in-memory caching, distributed caching, or response caching. Configure services in ConfigureServices using AddMemoryCache, AddDistributedRedisCache, or AddResponseCaching, and use attributes like [ResponseCache] or middleware like UseResponseCaching.

**64. What is the purpose of the Configure method in the Startup class?**

**Answer:** The Configure method defines the HTTP request pipeline using middleware components. It sets up how requests are handled and processed by the application, such as routing, authentication, and error handling.

**65. How do you handle file uploads in ASP.NET Core?**

**Answer:** Handle file uploads using the IFormFile interface in controller actions. Files can be read, saved, or processed within the action. Ensure the form is set to enctype="multipart/form-data".

**66. What is the role of the IHostEnvironment interface?**

**Answer:** IHostEnvironment provides information about the environment the application is running in, such as development, staging, or production. It offers properties for environment name, application root path, and web root path.

**67. How do you configure logging in ASP.NET Core?**

**Answer:** Configure logging using the ILogger interface. Set up logging providers in ConfigureServices using AddLogging and configure in appsettings.json. Providers include Console, Debug, and third-party options like Serilog.

**68. What is the difference between synchronous and asynchronous programming in .NET Core?**

**Answer:** Synchronous programming waits for tasks to complete before moving on, blocking the thread. Asynchronous programming uses async and await to allow tasks to run concurrently without blocking, improving responsiveness and performance.

**69. Explain middleware ordering in ASP.NET Core.**

**Answer:** Middleware components are executed in the order they are added in the Configure method. The order is crucial as it defines the sequence of request processing. Some middleware should be placed early, like error handling, while others, like static files, should be placed later.

**70. How do you create a custom middleware in ASP.NET Core?**

**Answer:** Create custom middleware by defining a class with an Invoke or InvokeAsync method. Register the middleware in the Configure method using app.UseMiddleware<CustomMiddleware>().  
  
**71. What is the purpose of appsettings.json in .NET Core?**

**Answer:** appsettings.json is a configuration file used to store application settings such as connection strings, API keys, and other environment-specific configurations. It is read during application startup to configure various services.

**72. How do you use Razor Pages in ASP.NET Core?**

**Answer:** Razor Pages is a page-based programming model for building web applications. Define Razor Pages in .cshtml files within the Pages folder, with a corresponding code-behind file for C# logic. Use AddRazorPages in ConfigureServices and MapRazorPages in Configure to enable routing.

**73. What is the purpose of Program.cs in a .NET Core application?**

**Answer:** Program.cs contains the Main method, which is the entry point of the application. It sets up and configures the web host or application host, defining how the application should be run.

**74. How do you handle database connections and context in Entity Framework Core?**

**Answer:** Define a DbContext class representing the database session and register it with dependency injection using AddDbContext in ConfigureServices. Configure connection strings in appsettings.json and use the context for querying and saving data.

**75. What is the UseRouting middleware used for in ASP.NET Core?**

**Answer:** UseRouting is used to enable endpoint routing in the middleware pipeline. It sets up routing to match incoming requests to endpoints defined in the application, such as controllers, Razor Pages, or minimal APIs.

**76. How do you configure custom routes in ASP.NET Core?**

**Answer:** Define custom routes using attributes like [Route] in controllers or actions. For more advanced routing, configure routes in Startup using the MapControllerRoute or MapDefaultControllerRoute methods in Configure.

**77. What is the role of the IConfiguration interface?**

**Answer:** IConfiguration provides a way to access application settings and configuration values from various sources like JSON files, environment variables, and command-line arguments. It is used to read and bind configuration settings.

**78. Explain the use of IApplicationBuilder in ASP.NET Core.**

**Answer:** IApplicationBuilder is used to configure the application's request pipeline. It allows you to add middleware components to handle requests, such as authentication, routing, and static files.

**79. How do you set up and use Health Checks in .NET Core?**

**Answer:** Add health check services using AddHealthChecks in ConfigureServices. Configure health check endpoints in Configure using MapHealthChecks. Health checks can monitor application and dependency health, such as database connectivity.

**80. What is the IServiceScope interface used for in .NET Core?**

**Answer:** IServiceScope is used to create a scope for resolving services with a Scoped lifetime. It is often used in background services or manually managed scopes to control the lifetime of scoped services within specific operations.

**81. What is IServiceProvider in .NET Core?**

**Answer:** IServiceProvider is an interface used to resolve services from the dependency injection container. It provides methods to get instances of services registered in the service collection.

**82. How do you use Custom Configuration Providers in .NET Core?**

**Answer:** Custom Configuration Providers allow you to load configuration from sources other than the default files. Implement IConfigurationProvider and IConfigurationSource interfaces, and add your provider using Add methods in ConfigureAppConfiguration.

**83. What is the purpose of IApplicationLifetime?**

**Answer:** IApplicationLifetime provides hooks for managing the application lifecycle, including handling application startup, shutdown, and stopping events. It allows you to run code during these lifecycle phases.

**84. How do you implement JWT Authentication in ASP.NET Core?**

**Answer:** Install the Microsoft.AspNetCore.Authentication.JwtBearer package. Configure JWT options in ConfigureServices using AddAuthentication and AddJwtBearer. Validate tokens and set up authorization policies.

**85. What is the UseAuthorization middleware used for?**

**Answer:** UseAuthorization middleware is used to enforce authorization policies and ensure that users have the required permissions to access specific resources or endpoints.

**86. How do you manage application secrets in .NET Core?**

**Answer:** Use the Secret Manager tool for local development to store sensitive data. For production, use environment variables or a secrets management service like Azure Key Vault.

**87. What is the IServiceCollection interface?**

**Answer:** IServiceCollection is a collection of service descriptors used to register services with the dependency injection container. It is configured in ConfigureServices to add, configure, or replace services.

**88. How do you implement Logging in .NET Core?**

**Answer:** Use the built-in logging framework by configuring logging providers like Console, Debug, and others in ConfigureServices. Use ILogger<T> in classes to log messages, which are then processed by the configured providers.

**89. What is the difference between UseMvc and UseEndpoints?**

**Answer:**

* UseMvc is used in earlier versions of ASP.NET Core for routing and MVC setup.
* UseEndpoints is part of the new routing system in ASP.NET Core 3.0 and later, enabling endpoint routing and middleware configuration.

**90. How do you perform a migration using Entity Framework Core?**

**Answer:** Use dotnet ef migrations add MigrationName to create a new migration and dotnet ef database update to apply migrations to the database.

**91. What is IDisposable in .NET Core?**

**Answer:** IDisposable is an interface used to release unmanaged resources explicitly. Implement the Dispose method to clean up resources like database connections, file handles, etc.

**92. How do you set up Dependency Injection for a custom service?**

**Answer:** Create a service class and register it in ConfigureServices using the appropriate lifetime method (AddTransient, AddScoped, AddSingleton).

**93. What are IActionFilter and IResultFilter in ASP.NET Core?**

**Answer:**

* IActionFilter allows you to run code before or after an action method executes.
* IResultFilter allows you to run code before or after the result of an action method is executed.

**94. How do you implement custom error pages in ASP.NET Core?**

**Answer:** Use UseExceptionHandler middleware to specify a custom error handling endpoint. Define error pages or handling logic in the specified endpoint.

**95. What is the IApplicationBuilder interface?**

**Answer:** IApplicationBuilder is used to configure the middleware pipeline for handling HTTP requests. It provides methods for adding middleware components to the pipeline.

**96. How do you use the HttpClient class in .NET Core?**

**Answer:** Use HttpClient to send HTTP requests and receive responses. Register HttpClient using AddHttpClient in ConfigureServices, and inject it into services or controllers.

**97. What is a Middleware in ASP.NET Core?**

**Answer:** Middleware is a component in the request pipeline that can handle requests, modify them, or perform actions before or after the request reaches the endpoint.

**98. How do you handle file downloads in ASP.NET Core?**

**Answer:** Use FileStreamResult or PhysicalFileResult to handle file downloads. Set the appropriate content type and file path or stream in the controller action.

**99. What is the purpose of the Startup class?**

**Answer:** The Startup class is used to configure the services and middleware in the application. It contains ConfigureServices for service registration and Configure for setting up the request pipeline.

**100. How do you enable and configure HTTPS redirection in ASP.NET Core?**

**Answer:** Use UseHttpsRedirection middleware in Configure to automatically redirect HTTP requests to HTTPS. Configure options in ConfigureServices if needed.

**101. What is the RequestDelegate in ASP.NET Core?**

**Answer:** RequestDelegate is a delegate that represents a method to process an HTTP request. It is used in middleware to handle requests and responses in the pipeline.

**102. How do you use Entity Framework Core with multiple databases?**

**Answer:** Configure multiple DbContext instances, each with its own connection string and settings. Register each DbContext in ConfigureServices and use them as needed.

**103. What is the purpose of IWebHostBuilder?**

**Answer:** IWebHostBuilder is used to configure and create a web host for running the application. It sets up the web server, configuration, and other services.

**104. How do you configure authentication in ASP.NET Core?**

**Answer:** Configure authentication by adding authentication services in ConfigureServices using AddAuthentication. Set up authentication schemes and options, then use UseAuthentication middleware in Configure.

**105. What is the HttpContext class?**

**Answer:** HttpContext provides information about the current HTTP request, such as request and response objects, user identity, and more. It is used to access request-specific data within controllers and middleware.

**106. How do you use IOptions<T> for configuration in .NET Core?**

**Answer:** Bind configuration settings to a class using IOptions<T>. Register the configuration in ConfigureServices using Configure<T>, and inject IOptions<T> into classes to access configuration values.

**107. What is the ConfigureAppConfiguration method used for?**

**Answer:** ConfigureAppConfiguration is used to set up and customize the application's configuration sources, such as JSON files, environment variables, or command-line arguments.

**108. How do you implement Authorization Policies in ASP.NET Core?**

**Answer:** Define authorization policies in ConfigureServices using AddAuthorization with AddPolicy. Apply policies using [Authorize(Policy = "PolicyName")] attribute on controllers or actions.

**109. What are Razor Pages and how do they differ from MVC?**

**Answer:** Razor Pages is a page-based programming model for building web applications, focusing on individual pages rather than controllers and views. It simplifies page handling and is more suitable for scenarios where individual pages are the primary focus.

**110. How do you configure application settings in different environments?**

**Answer:** Use environment-specific configuration files like appsettings.Development.json and appsettings.Production.json. The configuration system automatically loads the appropriate settings based on the environment.

**111. What is IConfigurationRoot in .NET Core?**

**Answer:** IConfigurationRoot is the interface representing the root of the configuration system. It provides access to the entire configuration tree and allows manipulation of configuration settings.

**112. How do you use IServiceScopeFactory in .NET Core?**

**Answer:** IServiceScopeFactory is used to create service scopes manually. It is typically used in background tasks or services that need to resolve scoped services.

**113. What is the HttpContext.Items collection used for?**

**Answer:** HttpContext.Items is a key-value collection used to store data that is specific to the current request. It allows sharing data between different middleware or components within the same request.

**114. How do you perform dependency injection in ASP.NET Core MVC views?**

**Answer:** Inject dependencies into views using the @inject directive in Razor views or by passing dependencies through view components or tag helpers.

**115. What is the UseAuthorization middleware in ASP.NET Core?**

**Answer:** UseAuthorization middleware is used to enforce authorization policies and ensure users have appropriate permissions to access specific resources.

**116. How do you use IServiceCollection to register services in .NET Core?**

**Answer:** Register services in IServiceCollection within ConfigureServices using methods like AddTransient, AddScoped, or AddSingleton to define the lifetime and behavior of services.

**117. What is DbContextOptions in Entity Framework Core?**

**Answer:** DbContextOptions is a class used to configure and provide options for a DbContext, such as connection strings, database providers, and other settings.

**118. How do you handle concurrency in Entity Framework Core?**

**Answer:** Handle concurrency using concurrency tokens, like a RowVersion property. Configure it using [ConcurrencyCheck] or [Timestamp] attributes and handle exceptions related to concurrency conflicts.

**119. What is the IActionResult interface?**

**Answer:** IActionResult is an interface used to represent the result of an action method. It allows returning different types of responses, such as views, JSON, or status codes.

**120. How do you implement Custom Model Binding in ASP.NET Core?**

**Answer:** Create a custom model binder by implementing IModelBinder and register it in ConfigureServices using ModelBinderProviders. It allows custom logic for binding model data from the request.

**121. What is IFileProvider and how is it used?**

**Answer:** IFileProvider is an interface used to access files and directories. It can be used to serve static files, access embedded resources, or customize file access.

**122. How do you use IHostedService for background tasks?**

**Answer:** Implement IHostedService to create background services. Register the service in ConfigureServices, and use StartAsync and StopAsync methods to manage background operations.

**123. What is the IModelMetadataProvider interface?**

**Answer:** IModelMetadataProvider is used to provide metadata about model properties, such as data types, validation rules, and display names, used in model binding and validation.

**124. How do you configure Swagger for API documentation?**

**Answer:** Install the Swashbuckle.AspNetCore package and add AddSwaggerGen in ConfigureServices. Configure Swagger in Configure using UseSwagger and UseSwaggerUI.

**125. What is IServiceProvider and how does it work?**

**Answer:** IServiceProvider is an interface used to resolve services from the dependency injection container. It provides methods to retrieve instances of registered services.

**126. How do you use the ActionFilterAttribute for custom filtering?**

**Answer:** Create a custom filter by inheriting from ActionFilterAttribute. Override methods like OnActionExecuting or OnActionExecuted to add custom logic before or after an action method.

**127. What is IOptionsSnapshot<T> used for?**

**Answer:** IOptionsSnapshot<T> provides access to configuration options with support for reloading options if the configuration changes. It is useful for retrieving options in a scoped manner.

**128. How do you implement Custom Authorization in ASP.NET Core?**

**Answer:** Create a custom authorization handler by implementing IAuthorizationHandler and defining policies using AddAuthorization in ConfigureServices. Apply the policies using [Authorize(Policy = "PolicyName")].

**129. What is IActionDescriptorCollectionProvider used for?**

**Answer:** IActionDescriptorCollectionProvider provides access to the collection of action descriptors, which represent the available actions and their metadata in the application.

**130. How do you use the UseStaticFiles middleware?**

**Answer:** UseStaticFiles middleware is used to serve static files (like HTML, CSS, and JS) from the wwwroot folder. It should be added early in the middleware pipeline.

**131. What are Configuration Providers and how do you use them?**

**Answer:** Configuration Providers are components that supply configuration data from different sources (files, environment variables, etc.). Use them by adding providers in ConfigureAppConfiguration during application startup.

**132. How do you implement Custom Validation in ASP.NET Core?**

**Answer:** Create a custom validation attribute by inheriting from ValidationAttribute. Override the IsValid method to define custom validation logic and apply it to model properties.

**133. What is HttpClientFactory and how is it used?**

**Answer:** HttpClientFactory is used to create and manage HttpClient instances, promoting reuse, improving performance, and handling DNS changes. Configure it in ConfigureServices and inject HttpClient where needed.

**134. What is the purpose of IApplicationBuilder in ASP.NET Core?**

**Answer:** IApplicationBuilder is used to configure the application's request pipeline by adding middleware components that process HTTP requests and responses.

**135. How do you use Dependency Injection in ASP.NET Core?**

**Answer:** Configure services in ConfigureServices using AddTransient, AddScoped, or AddSingleton. Inject services into constructors or methods where needed throughout the application.

**136. What is IWebHostEnvironment and how is it used?**

**Answer:** IWebHostEnvironment provides information about the hosting environment, such as environment name, application root, and web root paths. It helps to configure services or behaviors based on the environment.

**137. How do you implement Rate Limiting in ASP.NET Core?**

**Answer:** Implement rate limiting using middleware or third-party libraries. Configure rules for limiting the number of requests per time window and apply these rules globally or to specific endpoints.

**138. What are View Components and how are they different from Partial Views?**

**Answer:** View Components are reusable components that encapsulate rendering logic and are invoked from views or other components. Unlike partial views, they have their own view model and do not use a controller.

**139. How do you manage application settings in .NET Core?**

**Answer:** Use appsettings.json for default settings and environment-specific files like appsettings.Development.json for environment-specific settings. Access settings using IConfiguration.

**140. What is the role of IApplicationBuilder.Use() method?**

**Answer:** The Use() method is used to add middleware components to the request pipeline. It allows for custom middleware logic to be executed for each request.

**141. What is the IActionContextAccessor used for?**

**Answer:** IActionContextAccessor provides access to the current ActionContext, which includes details about the HTTP request, action method, and model binding. It is useful for accessing request-specific information in services.

**142. How do you implement IModelBinder for custom model binding?**

**Answer:** Implement IModelBinder to create custom logic for binding model data from the request. Register the custom model binder in ConfigureServices using ModelBinderProviders.

**143. What is IApplicationBuilder.UseWhen() used for?**

**Answer:** UseWhen() allows you to conditionally add middleware based on the request criteria. It enables you to apply middleware only for specific requests or request paths.

**144. How do you enable and configure CORS in ASP.NET Core?**

**Answer:** Add CORS services using AddCors in ConfigureServices. Configure policies with allowed origins, methods, and headers. Apply the policy using UseCors middleware in Configure.

**145. What is IWebHostBuilder used for in ASP.NET Core?**

**Answer:** IWebHostBuilder is used to configure and build the web host for running the application. It sets up server options, configuration, and dependency injection.

**146. How do you handle exceptions globally in ASP.NET Core?**

**Answer:** Use UseExceptionHandler middleware to configure a global error handling endpoint. You can also use UseDeveloperExceptionPage for detailed error information in development.

**147. What are Tag Helpers in ASP.NET Core?**

**Answer:** Tag Helpers are a way to add server-side logic to HTML elements in Razor views. They enable server-side code to dynamically generate HTML attributes and content.

**148. How do you implement Localization in ASP.NET Core?**

**Answer:** Add localization services using AddLocalization in ConfigureServices. Use resource files (.resx) for translations and apply localization to controllers or views using IStringLocalizer or IHtmlLocalizer.

**149. What is IServiceScope and when is it used?**

**Answer:** IServiceScope is used to create a scope for resolving services with Scoped lifetime. It is useful for managing the lifecycle of scoped services in scenarios like background tasks or manual scope creation.

**150. How do you use IHostingEnvironment for environment-specific configurations?**

**Answer:** IHostingEnvironment provides information about the hosting environment. Use it to conditionally load configuration settings, enable or disable features, or apply environment-specific logic.

**151. What is IHttpContextAccessor and how is it used?**

**Answer:** IHttpContextAccessor provides access to the current HttpContext from services. It is used to retrieve request-specific information or manage user sessions in non-controller classes.

**152. How do you set up and use IHostedService for background tasks?**

**Answer:** Implement IHostedService to define background tasks. Register the service in ConfigureServices using AddHostedService, and use StartAsync and StopAsync for task management.

**153. What is the role of RequestDelegate in ASP.NET Core?**

**Answer:** RequestDelegate is a delegate that processes HTTP requests. It is used in middleware to handle requests and responses in the request pipeline.

**154. How do you handle data validation in ASP.NET Core?**

**Answer:** Use data annotations on model properties for validation rules. Implement custom validation attributes or use FluentValidation for more complex validation scenarios. Validation results are processed automatically during model binding.

**155. What is the purpose of IModelBinderProvider?**

**Answer:** IModelBinderProvider is used to provide custom model binders for specific model types or binding scenarios. It allows for custom logic when binding model data from the request.

**156. How do you use ActionDescriptor in ASP.NET Core?**

**Answer:** ActionDescriptor represents metadata about an action method, such as route information and parameters. It is used internally by the MVC framework for routing and action execution.

**157. What is IActionResult<T> and how is it different from IActionResult?**

**Answer:** IActionResult<T> is a generic version of IActionResult that allows returning typed results, such as ActionResult<T> for APIs. It provides a more specific return type compared to the non-generic IActionResult.

**158. How do you use IConfigurationSection to access nested configuration values?**

**Answer:** Use IConfigurationSection to access nested configuration values by calling GetSection("SectionName") on IConfiguration. Bind the section to a POCO class or access values directly.

**159. What is IEndpointRouteBuilder and how is it used?**

**Answer:** IEndpointRouteBuilder is used to define and configure endpoints in the routing system. It allows adding routes for controllers, Razor Pages, and other endpoints in the request pipeline.

**160. How do you handle large file uploads in ASP.NET Core?**

**Answer:** Handle large file uploads by configuring request limits using KestrelServerOptions or IISServerOptions. Use streaming to process large files and avoid memory issues.

**161. What are Custom Middleware components in ASP.NET Core?**

**Answer:** Custom middleware components are user-defined components that handle HTTP requests and responses. They are implemented as classes with an Invoke or InvokeAsync method and added to the request pipeline.

**162. How do you use IModelValidator for custom validation?**

**Answer:** Implement IModelValidator to create custom validation logic for models. Register the validator in ConfigureServices, and it will be used during model validation.

**163. What is IModelValidatorProvider used for?**

**Answer:** IModelValidatorProvider is used to provide custom validators for model types. It allows defining how validation should be applied based on the model’s metadata.

**164. How do you implement Custom View Components in ASP.NET Core?**

**Answer:** Create a view component by inheriting from ViewComponent. Implement the Invoke or InvokeAsync method to provide logic and render a view associated with the component.

**165. What is IOptionsMonitor<T> and how does it differ from IOptionsSnapshot<T>?**

**Answer:** IOptionsMonitor<T> provides access to configuration options with support for change notifications, allowing you to react to changes. IOptionsSnapshot<T> provides access to options with support for reloading but does not support change notifications.

**166. How do you use IWebHostBuilder.ConfigureAppConfiguration()?**

**Answer:** ConfigureAppConfiguration() is used to add or modify configuration sources when building the web host. It allows configuring how the application reads and processes configuration settings.

**167. What is the role of IApplicationLifetime in managing application shutdown?**

**Answer:** IApplicationLifetime provides events and methods for managing the application lifecycle, including graceful shutdown. It allows running cleanup code during application stopping and stopped events.

**168. How do you use IHostingEnvironment to manage environment-specific settings?**

**Answer:** IHostingEnvironment provides properties for accessing environment information, such as environment name. Use it to conditionally load settings or configure services based on the current environment.

**169. How do you implement Custom Error Handling middleware in ASP.NET Core?**

**Answer:** Create custom error handling middleware by defining a class with an Invoke or InvokeAsync method. Add it to the middleware pipeline using UseMiddleware<CustomErrorHandlingMiddleware>().

**170. What is IMemoryCache and how is it used?**

**Answer:** IMemoryCache is used to cache data in memory. It provides methods to store, retrieve, and remove cache entries. Configure caching in ConfigureServices and use it to improve performance by reducing repeated data retrieval.

**171. How do you use IServiceProvider.GetRequiredService()?**

**Answer:** GetRequiredService<T>() retrieves a service from the dependency injection container. It throws an exception if the service is not registered, ensuring that the required service is available.

**172. What is IActionResult<T> and how does it benefit API development?**

**Answer:** IActionResult<T> is used in API development to return typed results. It provides a way to return strongly-typed data along with HTTP status codes, enhancing the clarity and type safety of API responses.

**173. How do you use IServiceCollection to configure scoped services?**

**Answer:** Register scoped services in IServiceCollection using AddScoped<TService, TImplementation>. Scoped services are created once per request or per scope, allowing for request-specific data management.

**174. What is IContentRootFileProvider and how is it used?**

**Answer:** IContentRootFileProvider is used to access files located in the application’s content root directory. It is useful for accessing files that are part of the application’s deployment package.

**175. How do you handle Authorization in ASP.NET Core?**

**Answer:** Handle authorization by configuring policies and roles using AddAuthorization in ConfigureServices. Apply authorization policies using [Authorize] attributes on controllers or actions.

**176. What is IApplicationBuilder.UseEndpoints() used for?**

**Answer:** UseEndpoints() is used to define and configure endpoint routing in ASP.NET Core. It allows you to map routes to controllers, Razor Pages, or other endpoints.

**177. How do you configure IHttpClientFactory for named clients?**

**Answer:** Configure named clients in ConfigureServices using AddHttpClient("ClientName", client => { ... }). It allows you to set up different configurations for multiple HttpClient instances.

**178. What are HTTP Middleware components and how are they used?**

**Answer:** HTTP middleware components are classes that handle HTTP requests and responses. They are used to add functionality such as authentication, logging, or custom processing in the request pipeline.

**179. How do you use IServiceCollection to register singleton services?**

**Answer:** Register singleton services in IServiceCollection using AddSingleton<TService, TImplementation>. Singleton services are created once and shared throughout the application’s lifetime.

**180. What is the role of IDesignTimeDbContextFactory<TContext> in Entity Framework Core?**

**Answer:** IDesignTimeDbContextFactory<TContext> is used to create instances of DbContext at design time. It is useful for tools like migrations and scaffolding, especially when the DbContext requires special configuration.

**181. What is IMiddleware and how is it used in ASP.NET Core?**

**Answer:** IMiddleware is an interface used to define custom middleware components. Implement the InvokeAsync method and add the middleware to the request pipeline using UseMiddleware<CustomMiddleware>().

**182. How do you configure Entity Framework Core to use a SQL Server database?**

**Answer:** Use optionsBuilder.UseSqlServer(connectionString) in DbContext to configure Entity Framework Core to use SQL Server. Configure the connection string in ConfigureServices using AddDbContext.

**183. What is IConfigurationBuilder and how is it used?**

**Answer:** IConfigurationBuilder is used to build and configure the IConfiguration object. It allows adding configuration sources, like JSON files or environment variables, and building the final configuration.

**184. How do you implement and use Custom Routing in ASP.NET Core?**

**Answer:** Implement custom routing by defining routes in Configure using app.UseEndpoints or app.UseRouter. Create route templates and map them to specific controllers or handlers.

**185. What is IServiceCollection and how is it used?**

**Answer:** IServiceCollection is used to register services with the dependency injection container. It allows configuring service lifetimes and dependencies using methods like AddTransient, AddScoped, and AddSingleton.

**186. How do you use IActionResult to return JSON data in an API controller?**

**Answer:** Return JSON data using Ok(object data) or Json(object data) methods of IActionResult in API controllers. These methods serialize the object to JSON format and set the appropriate content type.

**187. What is IServiceCollection.AddControllers() used for?**

**Answer:** AddControllers() is used to add MVC services to the DI container, enabling the use of controllers and related features in ASP.NET Core applications.

**188. How do you implement Custom Model Binding in ASP.NET Core?**

**Answer:** Create a custom model binder by implementing IModelBinder. Register the custom binder in ConfigureServices using ModelBinderProviders, and apply it to model properties or types.

**189. What is IActionDescriptorCollectionProvider and how is it used?**

**Answer:** IActionDescriptorCollectionProvider provides access to the collection of action descriptors. It is used for querying metadata about actions and routes in the application.

**190. How do you use IOptions<T> to manage configuration settings?**

**Answer:** Use IOptions<T> to access configuration settings by binding configuration data to a POCO class. Configure options in ConfigureServices using Configure<T> and access them via dependency injection.

**191. What is IServiceScopeFactory and how is it used?**

**Answer:** IServiceScopeFactory is used to create service scopes manually. It is typically used to resolve scoped services outside of request handling, like in background tasks or initialization code.

**192. How do you implement Custom Authorization Policies in ASP.NET Core?**

**Answer:** Define custom authorization policies using AddAuthorization in ConfigureServices. Implement a custom IAuthorizationHandler and register it to handle policy requirements.

**193. What is the IHttpClientFactory and how is it used for named clients?**

**Answer:** IHttpClientFactory provides a way to create HttpClient instances with specific configurations. Register named clients in ConfigureServices and use CreateClient("ClientName") to retrieve them.

**194. How do you configure Kestrel to limit request body size?**

**Answer:** Configure request body size limits in KestrelServerOptions using options.Limits.MaxRequestBodySize. Set this in ConfigureServices or Configure to control the maximum size of request bodies.

**195. What is IFileProvider and how is it used to serve static files?**

**Answer:** IFileProvider is an interface for accessing file systems. Use it to serve static files by configuring StaticFileOptions.FileProvider and setting it in UseStaticFiles middleware.

**196. How do you implement Custom Exception Handling in ASP.NET Core?**

**Answer:** Create custom exception handling middleware by implementing a class with InvokeAsync. Use UseMiddleware<CustomExceptionHandlingMiddleware>() in the request pipeline to handle exceptions.

**197. What is IWebHostEnvironment and how is it different from IHostingEnvironment?**

**Answer:** IWebHostEnvironment provides environment-specific information for the web host, including content root and web root paths. It is an updated version of IHostingEnvironment in newer ASP.NET Core versions.

**198. How do you configure Entity Framework Core for database migrations?**

**Answer:** Use AddDbContext to configure EF Core. Run Add-Migration MigrationName and Update-Database commands in the Package Manager Console to create and apply migrations.

**199. What is IApplicationBuilder.UseRouting() used for?**

**Answer:** UseRouting() is used to enable endpoint routing in ASP.NET Core. It sets up routing for endpoints like controllers, Razor Pages, and other route handlers.

**200. How do you use IConfigurationRoot to manage application settings?**

**Answer:** IConfigurationRoot represents the root of the configuration system. Use it to access and manipulate configuration settings from various sources like JSON files, environment variables, or command-line arguments.

**201. What is IOptionsMonitor<T> and how does it support options change tracking?**

**Answer:** IOptionsMonitor<T> provides access to options and tracks changes in real-time. It supports notifications when options are updated, allowing dynamic configuration updates.

**202. How do you implement Custom Endpoint Routing in ASP.NET Core?**

**Answer:** Define custom endpoint routing by configuring routes using Map methods in UseEndpoints. Create route templates and map them to controllers or handlers.

**203. What is IWebHostBuilder.ConfigureServices() and how is it used?**

**Answer:** ConfigureServices() allows you to configure services for dependency injection. It is used to register application services, middleware, and other components needed for the application.

**204. How do you use IServiceCollection.AddDbContext() to configure EF Core?**

**Answer:** Use AddDbContext<TContext>(options => options.UseSqlServer(connectionString)) to configure Entity Framework Core with a specific database provider and connection string.

**205. What is IActionContextAccessor and how can it be used for request-specific data?**

**Answer:** IActionContextAccessor provides access to the current ActionContext, which includes HTTP request and response details. It is used to retrieve request-specific data outside of controllers.

**206. How do you handle Concurrency Conflicts in Entity Framework Core?**

**Answer:** Handle concurrency conflicts by using concurrency tokens like RowVersion. Catch DbUpdateConcurrencyException and resolve conflicts by merging changes or applying retry logic.

**207. What is IFileProvider and how is it used in file access scenarios?**

**Answer:** IFileProvider provides an abstraction for file system access. It is used to access files in different locations, including the file system, embedded resources, and virtual file systems.

**208. How do you implement Custom Exception Filters in ASP.NET Core?**

**Answer:** Create a custom exception filter by implementing IExceptionFilter or IAsyncExceptionFilter. Register the filter in ConfigureServices and apply it globally or to specific controllers/actions.

**209. What is IApplicationBuilder.UseAuthorization() and how is it used?**

**Answer:** UseAuthorization() is used to add authorization middleware to the request pipeline. It ensures that authorization policies are enforced before processing requests.

**210. How do you configure ASP.NET Core to use In-Memory caching?**

**Answer:** Add in-memory caching services using AddMemoryCache in ConfigureServices. Use IMemoryCache to store and retrieve data from the cache within your application.

**211. What is IApplicationBuilder.UseEndpoints() and how is it used for routing?**

**Answer:** UseEndpoints() is used to configure and map endpoints to request handlers. It enables routing for controllers, Razor Pages, and other endpoints.

**212. How do you use IHttpContextAccessor to access HttpContext in a service?**

**Answer:** Inject IHttpContextAccessor into the service constructor to access the current HttpContext. Use HttpContextAccessor.HttpContext to retrieve request-specific information.

**213. What is IApplicationBuilder.UseRouting() used for in ASP.NET Core?**

**Answer:** UseRouting() is used to enable routing middleware. It sets up the routing system to map incoming requests to specific endpoints based on the route configuration.

**214. How do you configure Entity Framework Core to use SQLite?**

**Answer:** Use optionsBuilder.UseSqlite(connectionString) in DbContext to configure Entity Framework Core to use SQLite. Configure the connection string and add SQLite services in ConfigureServices.

**215. What is IServiceCollection.AddControllersWithViews() used for?**

**Answer:** AddControllersWithViews() registers services for MVC controllers and views, enabling the use of MVC pattern with support for both Razor views and API controllers.

**216. How do you use IServiceProvider to resolve services in a non-controller class?**

**Answer:** Inject IServiceProvider into the class constructor to resolve services using serviceProvider.GetService<T>() or serviceProvider.GetRequiredService<T>().

**217. What is IConfigurationBinder and how is it used?**

**Answer:** IConfigurationBinder is used to bind configuration settings to strongly-typed objects. It provides methods to map configuration data to POCO classes.

**218. How do you implement Custom Model Validation in ASP.NET Core?**

**Answer:** Implement custom validation attributes by creating a class that inherits from ValidationAttribute. Use the custom attribute on model properties to enforce validation rules.

**219. What is IApplicationBuilder.UseAuthentication() used for?**

**Answer:** UseAuthentication() adds authentication middleware to the request pipeline. It ensures that user authentication is processed before authorization checks.

**220. How do you configure ASP.NET Core to use Redis for distributed caching?**

**Answer:** Add Redis caching services using AddStackExchangeRedisCache in ConfigureServices. Configure Redis options with the connection string and use IDistributedCache for cache operations.

**221. What is IActionDescriptor and how does it contribute to routing?**

**Answer:** IActionDescriptor contains metadata about an action method, including route information, parameters, and filters. It is used by the routing system to match requests to actions.

**222. How do you configure Entity Framework Core for PostgreSQL?**

**Answer:** Use optionsBuilder.UseNpgsql(connectionString) in DbContext to configure Entity Framework Core to use PostgreSQL. Add the necessary NuGet package and configure the connection string.

**223. What is IApplicationBuilder.UseStaticFiles() used for?**

**Answer:** UseStaticFiles() adds middleware to serve static files from the web root directory. It allows access to files like images, CSS, and JavaScript in the application.

**224. How do you handle Custom Dependency Injection in ASP.NET Core?**

**Answer:** Define custom services and interfaces. Register them in IServiceCollection with appropriate lifetimes (transient, scoped, singleton) using methods like AddTransient, AddScoped, and AddSingleton.

**225. What is IOptions<T> and how do you use it to access configuration settings?**

**Answer:** IOptions<T> provides access to configuration settings by binding them to a POCO class. Configure options in ConfigureServices and access them through dependency injection.

**226. How do you use IApplicationBuilder.UseAuthorization() for authorization?**

**Answer:** UseAuthorization() adds authorization middleware to the pipeline. It ensures that requests are authorized according to the policies defined in ConfigureServices.

**227. What is IWebHostBuilder.UseStartup() and how is it used?**

**Answer:** UseStartup<Startup>() configures the application startup class. It is used to specify the Startup class for setting up services and middleware in the web host configuration.

**228. How do you implement Custom Middleware for logging requests?**

**Answer:** Create a middleware class with an InvokeAsync method to log request information. Use app.UseMiddleware<LoggingMiddleware>() to add the logging middleware to the request pipeline.

**229. What is IServiceScope and how is it used in ASP.NET Core?**

**Answer:** IServiceScope is used to create a scope for resolving services with Scoped lifetime. It allows managing the lifetime of scoped services manually, often used in background tasks.

**230. How do you use IConfiguration.GetValue<T>() to retrieve configuration settings?**

**Answer:** Use IConfiguration.GetValue<T>("Key") to retrieve configuration values by key. This method supports type conversion and is useful for accessing individual settings from the configuration.

**231. What is the purpose of IApplicationBuilder.UseEndpoints() in ASP.NET Core?**

**Answer:** UseEndpoints() is used to configure and map endpoints to request handlers. It sets up the routing for controllers, Razor Pages, and other endpoint types.

**232. How do you configure a custom logging provider in ASP.NET Core?**

**Answer:** Create a custom logging provider by implementing ILoggerProvider. Register it in ConfigureServices using AddProvider(new CustomLoggerProvider()).

**233. What is IConfiguration and how do you use it to manage application settings?**

**Answer:** IConfiguration is used to access configuration settings from various sources (e.g., JSON files, environment variables). Use it to read configuration values and bind them to POCO classes.

**234. How do you use IOptionsSnapshot<T> to handle configuration settings?**

**Answer:** IOptionsSnapshot<T> is used for accessing configuration settings that support change tracking. It provides access to configuration values and allows you to handle updates.

**235. What is IServiceProvider and how is it used to resolve services?**

**Answer:** IServiceProvider is an interface used to resolve services from the DI container. Use GetService<T>() or GetRequiredService<T>() to retrieve service instances.

**236. How do you implement Custom Model Binding in ASP.NET Core?**

**Answer:** Create a custom model binder by implementing IModelBinder. Register it in ConfigureServices and apply it to model properties or types using ModelBinderAttribute.

**237. What is IApplicationBuilder.UseAuthentication() used for in ASP.NET Core?**

**Answer:** UseAuthentication() adds authentication middleware to the request pipeline. It ensures that user authentication is processed before authorization checks are performed.

**238. How do you handle Data Annotations for model validation in ASP.NET Core?**

**Answer:** Use data annotations like [Required], [StringLength], and [Range] on model properties to enforce validation rules. ASP.NET Core automatically validates models based on these annotations.

**239. What is IServiceCollection.AddTransient() used for?**

**Answer:** AddTransient<TService, TImplementation>() registers a service with a transient lifetime. A new instance is created each time the service is requested.

**240. How do you implement Custom Action Filters in ASP.NET Core?**

**Answer:** Create a custom action filter by implementing IActionFilter or IAsyncActionFilter. Register the filter in ConfigureServices and apply it to controllers or actions.

**241. What is IApplicationBuilder.UseStaticFiles() used for?**

**Answer:** UseStaticFiles() adds middleware to serve static files (e.g., images, CSS, JavaScript) from the web root directory. It allows direct access to static resources.

**242. How do you configure ASP.NET Core to use JWT authentication?**

**Answer:** Configure JWT authentication using AddAuthentication().AddJwtBearer(). Set up options such as Authority and Audience in ConfigureServices, and add UseAuthentication() middleware.

**243. What is IServiceCollection.AddScoped() used for?**

**Answer:** AddScoped<TService, TImplementation>() registers a service with a scoped lifetime. A new instance is created per request but shared within the same request.

**244. How do you implement Custom Exception Handling middleware in ASP.NET Core?**

**Answer:** Create a custom middleware class with an InvokeAsync method to handle exceptions. Register the middleware in Configure using app.UseMiddleware<CustomExceptionHandlingMiddleware>().

**245. What is IWebHostEnvironment and how is it different from IHostingEnvironment?**

**Answer:** IWebHostEnvironment provides environment-specific information for the web host, such as content root and web root paths. It is the updated version of IHostingEnvironment in newer versions of ASP.NET Core.

**246. How do you use IHttpClientFactory to configure HttpClient with policies?**

**Answer:** Configure HttpClient with policies using AddHttpClient("ClientName").AddPolicyHandler() in ConfigureServices. Define policies like retry or circuit breaker to handle HTTP requests.

**247. What is IApplicationBuilder.UseRouting() used for?**

**Answer:** UseRouting() sets up the routing middleware, which enables endpoint routing by matching incoming requests to defined routes and endpoints.

**248. How do you implement Custom Middleware for request logging?**

**Answer:** Create a middleware class that logs request details in the InvokeAsync method. Register it using app.UseMiddleware<LoggingMiddleware>() in the request pipeline.

**249. What is IDesignTimeDbContextFactory<TContext> used for?**

**Answer:** IDesignTimeDbContextFactory<TContext> is used to create DbContext instances at design time for tasks like migrations or scaffolding, especially when the DbContext requires special setup.

**250. How do you use IOptions<T> to bind configuration sections?**

**Answer:** Use IOptions<T> to bind configuration sections to POCO classes. Configure options in ConfigureServices with Configure<T>(configuration.GetSection("SectionName")).

**251. What is IHttpContextAccessor and how is it used?**

**Answer:** IHttpContextAccessor provides access to the current HttpContext from classes that are not controllers. It is used to retrieve request-specific information in services.

**252. How do you configure ASP.NET Core for Redis caching?**

**Answer:** Add Redis caching services with AddStackExchangeRedisCache in ConfigureServices. Set Redis options like the connection string and use IDistributedCache to interact with Redis.

**253. What is IServiceCollection.AddSingleton() used for?**

**Answer:** AddSingleton<TService, TImplementation>() registers a service with a singleton lifetime. A single instance is created and shared throughout the application’s lifetime.

**254. How do you handle Concurrency in Entity Framework Core?**

**Answer:** Handle concurrency using concurrency tokens like RowVersion. Catch DbUpdateConcurrencyException and resolve conflicts by merging changes or applying retry logic.

**255. What is IWebHostBuilder.UseContentRoot() used for?**

**Answer:** UseContentRoot() sets the root directory for content files in the application. It specifies the base path for file access and is useful for configuration and file serving.

**256. How do you implement Custom Filters in ASP.NET Core?**

**Answer:** Create custom filters by implementing interfaces like IActionFilter, IAsyncActionFilter, or IResultFilter. Register them in ConfigureServices or apply them directly to controllers or actions.

**257. What is IApplicationBuilder.UseAuthorization() used for?**

**Answer:** UseAuthorization() adds authorization middleware to the request pipeline. It ensures that authorization policies are applied to requests before they are processed.

**258. How do you use IConfigurationRoot.GetSection() to access nested configuration settings?**

**Answer:** Use GetSection("Parent:Child") to access nested configuration settings. Bind the section to a POCO class or read individual values from the section.

**259. What is IServiceScopeFactory and how is it used for resolving scoped services?**

**Answer:** IServiceScopeFactory is used to create service scopes for resolving scoped services outside of request handling. It is often used in background tasks or initialization code.

**260. How do you implement Custom Model Validators in ASP.NET Core?**

**Answer:** Implement custom model validators by creating classes that inherit from IValidator<T>. Register them with the dependency injection container and apply them to model properties.

These additional questions should provide comprehensive coverage of various .NET Core topics for interview preparation.

4o mini

more

ChatGPT

Sure! Here are another 30 .NET Core interview questions and answers:

**261. What is IApplicationBuilder.UseEndpoints() used for in ASP.NET Core?**

**Answer:** UseEndpoints() is used to configure the endpoints of the application, such as MVC controllers, Razor Pages, and minimal APIs, and map them to incoming requests.

**262. How do you configure Entity Framework Core for SQL Server with connection string options?**

**Answer:** In ConfigureServices, use AddDbContext<MyDbContext>(options => options.UseSqlServer(configuration.GetConnectionString("DefaultConnection"))) to configure EF Core with SQL Server.

**263. What is IServiceCollection.AddControllers() used for?**

**Answer:** AddControllers() registers the services required for MVC controllers. It is used to enable support for API controllers without views.

**264. How do you implement Custom Authorization Attributes in ASP.NET Core?**

**Answer:** Create a custom authorization attribute by inheriting from AuthorizeAttribute and overriding the OnAuthorization method. Register the attribute and apply it to controllers or actions.

**265. What is IOptions<T> and how do you use it to manage configuration settings?**

**Answer:** IOptions<T> allows you to access configuration settings by binding them to a strongly-typed POCO class. Configure options in ConfigureServices and inject IOptions<T> into services.

**266. How do you configure ASP.NET Core to use Session State?**

**Answer:** Add session services with AddSession in ConfigureServices. Configure session options and add UseSession() in the middleware pipeline to manage session state.

**267. What is IApplicationBuilder.UseRouting() used for?**

**Answer:** UseRouting() sets up the routing middleware, which processes incoming requests and determines how they should be routed to endpoints based on the route configuration.

**268. How do you handle Global Exception Handling in ASP.NET Core?**

**Answer:** Implement global exception handling by using a middleware that catches exceptions in the InvokeAsync method and handles them appropriately. Register the middleware with UseMiddleware<ExceptionHandlingMiddleware>().

**269. What is IDistributedCache and how is it used in ASP.NET Core?**

**Answer:** IDistributedCache is an interface for distributed caching systems like Redis or SQL Server. Use it to store and retrieve cached data across different instances of the application.

**270. How do you configure ASP.NET Core to use Razor Pages?**

**Answer:** Add Razor Pages support using AddRazorPages() in ConfigureServices. Configure routing with UseEndpoints(endpoints => endpoints.MapRazorPages()) in the middleware pipeline.

**271. What is IServiceCollection.AddHealthChecks() used for?**

**Answer:** AddHealthChecks() adds health check services to the DI container. It is used to monitor the health of application dependencies and services.

**272. How do you implement Custom Routing in ASP.NET Core?**

**Answer:** Define custom routes in Configure using UseRouter or Map methods. Create route templates and map them to specific controllers or middleware.

**273. What is IApplicationBuilder.UseAuthentication() used for?**

**Answer:** UseAuthentication() adds authentication middleware to the request pipeline. It processes incoming requests to ensure that the user is authenticated.

**274. How do you use IWebHostBuilder to configure Kestrel settings?**

**Answer:** Configure Kestrel settings using ConfigureKestrel method on IWebHostBuilder. Set options like port, HTTPS, and request limits.

**275. What is IConfigurationRoot and how is it used to access configuration settings?**

**Answer:** IConfigurationRoot represents the root of the configuration system. Use it to access and manage configuration settings from various sources like JSON files and environment variables.

**276. How do you use IServiceCollection.AddScoped() to manage service lifetimes?**

**Answer:** AddScoped<TService, TImplementation>() registers a service with a scoped lifetime, creating a new instance per request but reusing it within the same request.

**277. What is IApplicationBuilder.UseStaticFiles() used for?**

**Answer:** UseStaticFiles() adds middleware to serve static files (e.g., CSS, JavaScript, images) from the web root directory, allowing direct access to these resources.

**278. How do you handle JSON Serialization in ASP.NET Core?**

**Answer:** Use System.Text.Json or Newtonsoft.Json to handle JSON serialization and deserialization. Configure the desired JSON serializer in ConfigureServices.

**279. What is IServiceScope and how is it used in dependency injection?**

**Answer:** IServiceScope creates a scope for resolving scoped services. It allows for managing the lifetime of scoped services manually, often used in background tasks or services.

**280. How do you configure ASP.NET Core to use Swagger for API documentation?**

**Answer:** Add Swagger services with AddSwaggerGen in ConfigureServices. Use UseSwagger and UseSwaggerUI in Configure to enable Swagger UI for API documentation.

**281. What is IApplicationBuilder.UseAuthorization() used for?**

**Answer:** UseAuthorization() adds authorization middleware to the pipeline. It ensures that authorization policies are enforced before the request is handled.

**282. How do you implement Custom Dependency Injection in ASP.NET Core?**

**Answer:** Define custom services and interfaces, then register them in ConfigureServices using AddTransient, AddScoped, or AddSingleton based on the desired lifetime.

**283. What is IApplicationBuilder.UseEndpoints() used for in ASP.NET Core?**

**Answer:** UseEndpoints() is used to configure the endpoints for the application, mapping routes to controllers, Razor Pages, or other endpoint types.

**284. How do you handle Configuration Reload in ASP.NET Core?**

**Answer:** Use IOptionsMonitor<T> to track changes in configuration settings. It supports real-time updates and can notify when configuration values change.

**285. What is IHttpContextAccessor and how do you use it to access HttpContext in non-controller classes?**

**Answer:** IHttpContextAccessor provides access to the current HttpContext from non-controller classes. Inject it into services to retrieve request-specific information.

**286. How do you implement Custom Exception Filters in ASP.NET Core?**

**Answer:** Create a custom exception filter by implementing IExceptionFilter or IAsyncExceptionFilter. Register it globally or apply it to specific controllers/actions.

**287. What is IWebHostEnvironment and how is it used to determine the hosting environment?**

**Answer:** IWebHostEnvironment provides information about the hosting environment, such as content root and environment name (Development, Staging, Production). Use it to adjust behavior based on the environment.

**288. How do you use IOptionsMonitor<T> to manage configuration changes?**

**Answer:** IOptionsMonitor<T> allows you to access configuration settings and subscribe to change notifications. Use it to handle dynamic updates to configuration values.

**289. What is IServiceCollection.AddHttpClient() used for?**

**Answer:** AddHttpClient() registers HttpClient instances with the DI container. It enables the configuration of HttpClient with policies, handlers, and default settings.

**290. How do you handle Caching in ASP.NET Core applications?**

**Answer:** Use caching services like IMemoryCache for in-memory caching and IDistributedCache for distributed caching. Configure and use these services to store and retrieve cached data.

These additional questions should help you cover even more aspects of .NET Core for interview preparation.

4o mini

more

ChatGPT

Certainly! Here are another 30 .NET Core interview questions and answers:

**291. What is IServiceCollection.AddMvc() used for?**

**Answer:** AddMvc() registers services required for MVC and Razor Pages support, including model binding, validation, and routing.

**292. How do you implement Custom Middleware to modify HTTP responses?**

**Answer:** Create a middleware class with an InvokeAsync method that modifies the HttpResponse. Register it in Configure using app.UseMiddleware<CustomResponseMiddleware>().

**293. What is IApplicationBuilder.UseEndpoints() and how is it different from UseRouting()?**

**Answer:** UseEndpoints() is used to configure endpoint routing, mapping routes to handlers. UseRouting() sets up the routing middleware, enabling request routing before endpoints are matched.

**294. How do you configure Entity Framework Core to use SQLite?**

**Answer:** Use optionsBuilder.UseSqlite(connectionString) in DbContext configuration. Add the SQLite NuGet package and configure the connection string in ConfigureServices.

**295. What is IServiceCollection.AddMemoryCache() used for?**

**Answer:** AddMemoryCache() registers the in-memory caching services. It allows you to use IMemoryCache to store and retrieve data in memory.

**296. How do you handle Logging in ASP.NET Core applications?**

**Answer:** Configure logging services using AddLogging in ConfigureServices. Use built-in providers or custom providers to log messages and configure logging levels.

**297. What is IConfigurationBuilder and how is it used?**

**Answer:** IConfigurationBuilder builds configuration from various sources, such as JSON files, environment variables, and command-line arguments. Use it to set up and configure IConfiguration.

**298. How do you implement Custom Model Binding in ASP.NET Core?**

**Answer:** Create a custom model binder by implementing IModelBinder and registering it in ConfigureServices. Use ModelBinderAttribute to apply it to specific properties or types.

**299. What is IServiceCollection.AddDbContext() used for?**

**Answer:** AddDbContext<TContext>() registers the DbContext with the dependency injection container. It configures the context for use with Entity Framework Core.

**300. How do you use IOptions<T> for strongly-typed configuration settings?**

**Answer:** Bind configuration sections to POCO classes using Configure<T>. Inject IOptions<T> into services to access and use the configuration settings.

**301. What is IWebHostBuilder.UseUrls() used for?**

**Answer:** UseUrls() configures the URLs that the web host listens on. It specifies the addresses and ports for the application to bind to.

**302. How do you implement Custom Data Protection in ASP.NET Core?**

**Answer:** Configure data protection using AddDataProtection in ConfigureServices. Implement custom data protection providers or keys if needed.

**303. What is IApplicationBuilder.UseSession() used for?**

**Answer:** UseSession() adds session middleware to the pipeline, allowing for session state management and storage of user-specific data across requests.

**304. How do you implement Custom Exception Handling using ExceptionHandlerMiddleware?**

**Answer:** Use app.UseExceptionHandler("/Home/Error") to redirect to a custom error handling page. Implement custom logic in the Error action of the Home controller.

**305. What is IWebHostBuilder.ConfigureAppConfiguration() used for?**

**Answer:** ConfigureAppConfiguration() is used to configure the application's configuration sources, such as JSON files, environment variables, and command-line arguments.

**306. How do you configure ASP.NET Core to use SignalR for real-time communication?**

**Answer:** Add SignalR services with AddSignalR in ConfigureServices. Use MapHub<YourHub>(endpoint) in Configure to set up SignalR hubs and endpoints.

**307. What is IApplicationBuilder.UseHttpsRedirection() used for?**

**Answer:** UseHttpsRedirection() adds middleware to redirect HTTP requests to HTTPS, ensuring secure communication between clients and the server.

**308. How do you configure ASP.NET Core for Dependency Injection with a custom service provider?**

**Answer:** Create and configure a custom service provider by implementing IServiceProvider. Register the custom provider in ConfigureServices and use it for resolving dependencies.

**309. What is IServiceCollection.AddHostedService() used for?**

**Answer:** AddHostedService<TService>() registers a hosted service that runs in the background, such as a background task or a long-running process.

**310. How do you implement Custom Configuration Providers in ASP.NET Core?**

**Answer:** Create a custom configuration provider by implementing ConfigurationProvider. Add it to the configuration builder using Add method.

**311. What is IApplicationBuilder.UseDeveloperExceptionPage() used for?**

**Answer:** UseDeveloperExceptionPage() adds middleware to display detailed exception information during development. It helps with debugging by showing stack traces and error details.

**312. How do you configure ASP.NET Core to use Azure Key Vault for configuration management?**

**Answer:** Add Azure Key Vault support with AddAzureKeyVault in ConfigureAppConfiguration. Provide the Key Vault client and configuration settings to access secrets.

**313. What is IApplicationBuilder.UseRouting() used for in ASP.NET Core?**

**Answer:** UseRouting() sets up the routing middleware to handle request routing based on route patterns and endpoint configurations.

**314. How do you implement Custom Cookie Authentication` in ASP.NET Core?**

**Answer:** Configure cookie authentication using AddAuthentication().AddCookie() in ConfigureServices. Set up options like Cookie.Name, Cookie.SecurePolicy, and Cookie.Expiration in the authentication configuration.

**315. What is IWebHostBuilder.ConfigureServices() used for?**

**Answer:** ConfigureServices() is used to add services to the dependency injection container. It allows configuration of application services, middleware, and options.

**316. How do you implement Custom Model Validators for complex validation scenarios?**

**Answer:** Create custom model validators by implementing IValidator<T>. Register validators in ConfigureServices and apply them to models or properties as needed.

**317. What is IApplicationBuilder.UseCookiePolicy() used for?**

**Answer:** UseCookiePolicy() adds middleware to handle cookie policy compliance, such as setting cookie policies and handling user consent.

**318. How do you use IApplicationBuilder.UseStaticFiles() for serving static files from a custom directory?**

**Answer:** Use UseStaticFiles(new StaticFileOptions { FileProvider = new PhysicalFileProvider(Path.Combine(Directory.GetCurrentDirectory(), "CustomStaticFiles")) }) to serve static files from a custom directory.

**319. What is IWebHostEnvironment.ContentRootPath and how is it used?**

**Answer:** ContentRootPath represents the root path of the application. It is used to access and manage files relative to the application's root directory.

**320. How do you configure ASP.NET Core for Custom URL Rewriting?**

**Answer:** Add URL rewriting services with AddRewriter in ConfigureServices. Use UseRewriter(new RewriteOptions().AddRewrite("oldPath", "newPath", skipRemainingRules: true)) to apply URL rewrite rules.

**321. What is IApplicationBuilder.UseAuthorization() used for in ASP.NET Core?**

**Answer:** UseAuthorization() adds middleware to enforce authorization policies, ensuring that users have the necessary permissions to access resources.

**322. How do you handle Database Migrations in Entity Framework Core?**

**Answer:** Use Add-Migration to create a migration script and Update-Database to apply the migration to the database. This ensures that your database schema matches your model.

**323. What is IServiceCollection.AddOptions() used for?**

**Answer:** AddOptions() registers services for configuration options. It allows binding configuration settings to strongly-typed classes and accessing them via IOptions<T>.

**324. How do you configure ASP.NET Core to use Cors for cross-origin requests?**

**Answer:** Use AddCors in ConfigureServices to set up CORS policies and UseCors in Configure to enable CORS middleware for cross-origin requests.

**325. What is IApplicationBuilder.UseAuthentication() used for?**

**Answer:** UseAuthentication() adds authentication middleware to the pipeline, which processes and validates user authentication for incoming requests.

**326. How do you use IServiceCollection.AddTransient() for dependency injection?**

**Answer:** AddTransient<TService, TImplementation>() registers a service with a transient lifetime, creating a new instance each time the service is requested.

**327. What is IWebHostBuilder.UseContentRoot() used for?**

**Answer:** UseContentRoot() sets the root directory for the application, which is used for locating files and configuring file paths.

**328. How do you implement Custom Middleware to log request information?**

**Answer:** Create a middleware class with an InvokeAsync method that logs request details. Register it in Configure using app.UseMiddleware<LoggingMiddleware>().

**329. What is IServiceCollection.AddSingleton() used for?**

**Answer:** AddSingleton<TService, TImplementation>() registers a service with a singleton lifetime, creating a single instance that is shared across the application.

**330. How do you implement Custom Authorization Policies` in ASP.NET Core?**

**Answer:** Define custom authorization policies using AddAuthorization in ConfigureServices. Implement IAuthorizationHandler for custom authorization logic.

**331. What is IApplicationBuilder.UseDeveloperExceptionPage() and when should it be used?**

**Answer:** UseDeveloperExceptionPage() provides detailed error information in development environments. It should be used only in development to help with debugging.

**332. How do you use IConfiguration to access configuration settings from appsettings.json?**

**Answer:** Inject IConfiguration into services or controllers to access configuration settings using configuration["Section:Key"] or bind settings to POCO classes.

**333. What is IServiceScopeFactory and how is it used?**

**Answer:** IServiceScopeFactory creates IServiceScope instances, allowing for the creation of scoped services manually. Useful for background tasks or services needing scoped instances.

**334. How do you configure ASP.NET Core to use Health Checks?**

**Answer:** Add health check services using AddHealthChecks in ConfigureServices. Use UseHealthChecks in Configure to expose health check endpoints.

**335. What is IApplicationBuilder.UseResponseCaching() used for?**

**Answer:** UseResponseCaching() adds middleware to cache HTTP responses, improving performance by reducing the need to generate responses repeatedly.

**336. How do you implement Custom Exception Handling using IExceptionFilter?**

**Answer:** Create a class implementing IExceptionFilter and override the OnException method to handle exceptions. Register the filter globally or for specific controllers.

**337. What is IWebHostBuilder.ConfigureLogging() used for?**

**Answer:** ConfigureLogging() allows configuring logging providers and settings, such as log levels and output destinations, for the application’s logging system.

**338. How do you use IApplicationBuilder.UseRouting() with MapControllers()?**

**Answer:** UseRouting() sets up the routing middleware, and MapControllers() configures the endpoints for MVC controllers. Both are needed to handle routing and endpoint mapping.

**339. What is IServiceCollection.AddDistributedMemoryCache() used for?**

**Answer:** AddDistributedMemoryCache() registers a distributed memory cache, which stores cache data in memory across multiple instances of the application.

**340. How do you configure ASP.NET Core to use Azure Blob Storage for file storage?**

**Answer:** Add Azure Blob Storage services with AddAzureBlobStorage in ConfigureServices. Configure the storage options and use BlobServiceClient to interact with Blob Storage.

**341. What is IApplicationBuilder.UseFileServer() used for?**

**Answer:** UseFileServer() combines the functionality of UseStaticFiles() and UseDefaultFiles() to serve static files and default files like index.html.

**342. How do you implement Custom Model Binding` for complex data structures?**

**Answer:** Create a custom model binder by implementing IModelBinder and register it with ModelBinderProviders. Use it to handle complex data structures in model binding.

**343. What is IWebHostBuilder.UseStartup() used for?**

**Answer:** UseStartup<TStartup>() configures the startup class that sets up services and the request pipeline for the application.

**344. How do you configure ASP.NET Core to use Redis for distributed caching?**

**Answer:** Add Redis services with AddStackExchangeRedisCache in ConfigureServices. Configure the Redis connection string and use IDistributedCache to interact with Redis.

**345. What is IApplicationBuilder.UseHttpContext() used for?**

**Answer:** UseHttpContext() is not a standard method. However, IHttpContextAccessor can be used to access HttpContext from services or middleware.

**346. How do you use IApplicationBuilder.UseExceptionHandler() for custom error pages?**

**Answer:** Use UseExceptionHandler("/Home/Error") to redirect exceptions to a custom error page. Configure the error handling logic in the specified error page or action.

**347. What is IServiceCollection.AddLogging() used for?**

**Answer:** AddLogging() registers logging services with the dependency injection container, allowing you to configure and use logging in the application.

**348. How do you implement Custom Model ValidationusingIValidator`?**

**Answer:** Create a custom validator by implementing IValidator<T>. Register it with the service collection and use it to validate models according to custom rules.

**349. What is IApplicationBuilder.UseSecurityHeaders() used for?**

**Answer:** UseSecurityHeaders() is not a built-in method. Security headers can be added using custom middleware or by configuring response headers in Configure.

**350. How do you configure ASP.NET Core to use Swagger with OAuth2 for authentication?**

**Answer:** Configure Swagger to use OAuth2 by adding authentication options in AddSwaggerGen and setting up OAuth2 security definitions and scopes. Use UseSwaggerUI to integrate with Swagger UI.

**351. What is IApplicationBuilder.UseAuthentication() and where should it be placed in the middleware pipeline?**

**Answer:** UseAuthentication() adds authentication middleware to the pipeline. It should be placed before UseAuthorization() to ensure authentication occurs before authorization checks.

**352. How do you configure ASP.NET Core to use NCache for distributed caching?**

**Answer:** Add NCache services with AddNCache in ConfigureServices. Configure the NCache settings and use IDistributedCache to interact with NCache.

**353. What is IConfigurationRoot and how is it different from IConfiguration?**

**Answer:** IConfigurationRoot is a concrete implementation of IConfiguration representing the root configuration provider. IConfiguration is an interface used for accessing configuration values.

**354. How do you set up ASP.NET Core to use Custom Dependency Injection containers?**

**Answer:** Use ServiceProvider to create and configure a custom DI container. Register services with the custom container and replace the default container if needed.

**355. What is IServiceCollection.AddHttpClient() used for?**

**Answer:** AddHttpClient() registers HttpClient services for dependency injection, allowing you to configure and use HttpClient instances with built-in features like retry policies.

**356. How do you handle Database Connection Resiliency in Entity Framework Core?**

**Answer:** Configure DbContext with connection resiliency options using EnableRetryOnFailure() to automatically retry failed database operations due to transient faults.

**357. What is IApplicationBuilder.UseStatusCodePages() used for?**

**Answer:** UseStatusCodePages() adds middleware to handle HTTP status codes and display status code pages or custom error responses.

**358. How do you use IApplicationBuilder.UseEndpoints() to configure route endpoints?**

**Answer:** UseEndpoints() maps endpoints to route handlers, such as MVC controllers or Razor Pages. Define routes and endpoints in the method parameter.

**359. What is IServiceCollection.AddSingleton() and how does it differ from AddTransient()?**

**Answer:** AddSingleton() registers a single instance of a service shared across the application, while AddTransient() creates a new instance each time the service is requested.

**360. How do you configure ASP.NET Core to use Custom Authentication schemes?**

**Answer:** Add custom authentication schemes using AddAuthentication().AddScheme<TOptions, THandler>() in ConfigureServices. Implement the authentication handler for custom logic.

**361. What is IServiceProvider and how is it used in ASP.NET Core?**

**Answer:** IServiceProvider is an interface for retrieving service instances. It is used to resolve services and manage dependencies throughout the application.

**362. How do you configure ASP.NET Core to use Health Checks with custom checks?**

**Answer:** Add custom health checks using AddHealthChecks().AddCheck<CustomHealthCheck>("custom") in ConfigureServices. Implement the custom health check class and logic.

**363. What is IApplicationBuilder.UseRouting() used for in relation to UseEndpoints()?**

**Answer:** UseRouting() sets up routing middleware to route requests to endpoints, while UseEndpoints() maps those routes to specific handlers or controllers.

**364. How do you implement Custom Exception Handling for API responses?**

**Answer:** Create a custom exception filter or middleware to handle exceptions and return custom API responses. Register it globally or for specific controllers.

**365. What is IConfigurationBuilder and how is it used to build configurations?**

**Answer:** IConfigurationBuilder builds configuration from multiple sources like JSON files, environment variables, and command-line arguments. It allows combining and configuring settings.

**366. How do you configure ASP.NET Core to use Custom Error Pages for different status codes?**

**Answer:** Use UseStatusCodePagesWithReExecute("/Error/{0}") to display custom error pages for different status codes, where {0} represents the status code.

**367. What is IServiceCollection.AddMemoryCache() and when should it be used?**

**Answer:** AddMemoryCache() registers in-memory caching services. It should be used when you need to cache data in memory to improve performance and reduce database load.

**368. How do you implement Custom Data Protection` in ASP.NET Core?**

**Answer:** Configure data protection services using AddDataProtection in ConfigureServices. Implement custom data protection providers or keys if required.

**369. What is IApplicationBuilder.UseStaticFiles() and how is it used to serve files?**

**Answer:** UseStaticFiles() adds middleware to serve static files from the wwwroot directory or a custom directory specified in StaticFileOptions.

**370. How do you configure ASP.NET Core to use Entity Framework Core with PostgreSQL?**

**Answer:** Add PostgreSQL support with AddDbContext using UseNpgsql(connectionString) in ConfigureServices. Install the Npgsql.EntityFrameworkCore.PostgreSQL package.

**371. What is IWebHostBuilder.UseKestrel() used for?**

**Answer:** UseKestrel() configures Kestrel as the web server for handling HTTP requests. It sets up the Kestrel server options and allows customization.

**372. How do you implement Custom Logging` in ASP.NET Core?**

**Answer:** Configure custom logging providers and settings in ConfigureLogging. Implement and register custom logging providers if needed.

**373. What is IServiceCollection.AddHostedService() and how is it used?**

**Answer:** AddHostedService() registers background services that run when the application starts. Implement IHostedService for long-running tasks.

**374. How do you use IApplicationBuilder.UseAuthorization() with Custom Authorization Policies?**

**Answer:** UseAuthorization() enforces custom authorization policies configured with AddAuthorization() in ConfigureServices. Apply policies to controllers or actions.

**375. What is IConfiguration.GetSection() and how is it used?**

**Answer:** GetSection() retrieves a specific section of the configuration, which can be bound to a POCO class or used to access nested configuration values.

**376. How do you configure ASP.NET Core for Custom Routing` with attributes?**

**Answer:** Use attribute routing by decorating controller actions with [Route("path")] or [HttpGet("path")]. Configure routing options in ConfigureServices.

**377. What is IApplicationBuilder.UseHttpsRedirection() used for?**

**Answer:** UseHttpsRedirection() adds middleware to automatically redirect HTTP requests to HTTPS, ensuring secure communication.

**378. How do you implement Custom Session Management` in ASP.NET Core?**

**Answer:** Configure session services with AddSession() in ConfigureServices. Use UseSession() in Configure to manage session state for user-specific data.

**379. What is IApplicationBuilder.UseEndpoints() and how does it interact with UseRouting()?**

**Answer:** UseEndpoints() maps request handlers to routes defined by UseRouting(), enabling endpoint routing and handling requests based on route patterns.

**380. How do you configure ASP.NET Core to use Custom Model Bindings`?**

**Answer:** Create a custom model binder by implementing IModelBinder. Register it with ModelBinderProviders and use it to bind complex or custom data types.

**381. What is IWebHostBuilder.UseConfiguration() used for?**

**Answer:** UseConfiguration() configures the web host with application settings and configuration sources, allowing customization of how configuration is loaded.

**382. How do you implement Custom Caching` strategies in ASP.NET Core?**

**Answer:** Use caching services like IMemoryCache or IDistributedCache to implement custom caching strategies. Configure cache options and policies according to application needs.

**383. What is IApplicationBuilder.UseFileServer() and how is it different from UseStaticFiles()?**

**Answer:** UseFileServer() combines static file serving and default file handling. It includes the functionality of UseStaticFiles() and UseDefaultFiles().

**384. How do you configure ASP.NET Core to use Custom HTTP Headers`?**

**Answer:** Add custom HTTP headers using middleware. Implement a middleware class that sets headers in the HttpContext.Response.Headers collection.

**385. What is IServiceCollection.AddControllers() used for?**

**Answer:** AddControllers() registers MVC services for controller-based APIs without views, focusing on RESTful API functionality.

**386. How do you implement Custom Background Services` in ASP.NET Core?**

**Answer:** Implement IHostedService for background tasks and register it using AddHostedService<TService>(). Define the background processing logic in the service class.

**387. What is IApplicationBuilder.UseSecurityHeaders() and how is it implemented?**

**Answer:** UseSecurityHeaders() is not a built-in method. Implement security headers by adding middleware to set headers like Content-Security-Policy and X-Content-Type-Options.

**388. How do you configure ASP.NET Core to use Custom Dependency Injection Scopes`?**

**Answer:** Define custom scopes using IServiceScopeFactory to create IServiceScope instances. Manage scoped services within the scope context.

**389. What is IConfigurationRoot.GetValue<T>() used for?**

**Answer:** GetValue<T>() retrieves a configuration value of type T from the configuration source, allowing you to access typed configuration settings.

**390. How do you use IApplicationBuilder.UseWelcomePage()?**

**Answer:** UseWelcomePage() adds middleware to display a simple welcome page for the application, useful for testing and initial setup.

**391. What is IServiceCollection.AddDataProtection() used for?**

**Answer:** AddDataProtection() registers data protection services for securing data, such as encrypting cookies and protecting sensitive information.

**392. How do you implement Custom Authentication Providers` in ASP.NET Core?**

**Answer:** Add custom authentication providers using AddAuthentication().AddScheme<TOptions, THandler>(). Implement the custom authentication handler for your provider.

**393. What is IApplicationBuilder.UseSerilog() and how is it used?**

**Answer:** UseSerilog() integrates Serilog logging with ASP.NET Core. Configure Serilog as the logging provider and use it to handle logging in your application.

**394. How do you configure ASP.NET Core for Custom User Claims`?**

**Answer:** Add custom claims to user identity in authentication handlers or middleware. Access and use claims for authorization and user-specific logic.

**395. What is IServiceCollection.AddMvcCore() and how is it different from AddMvc()?**

**Answer:** AddMvcCore() registers the core MVC services without features like views, while AddMvc() includes all MVC features including views, Razor pages, and more.

**396. How do you implement Custom Middleware` to modify request and response?**

**Answer:** Create a middleware class with InvokeAsync method to process requests and responses. Register it in the Configure method using app.UseMiddleware<YourMiddleware>().

**397. What is IApplicationBuilder.UseHealthChecks() used for?**

**Answer:** UseHealthChecks() adds middleware to expose health check endpoints for monitoring application health status.

**398. How do you configure ASP.NET Core to use Custom Authorization Filters`?**

**Answer:** Create custom authorization filters by implementing IAuthorizationFilter or IAsyncAuthorizationFilter. Register the filters globally or for specific actions.

**399. What is IServiceCollection.AddSignalR() used for?**

**Answer:** AddSignalR() registers SignalR services, enabling real-time communication features like hubs and persistent connections in the application.

**400. How do you configure ASP.NET Core to use Custom Options Validation`?**

**Answer:** Implement custom validation logic by defining IValidateOptions<T> and configuring options validation in ConfigureServices using services.AddOptions<T>().Validate.

**401. What is IApplicationBuilder.UseRouting() and how does it work with UseEndpoints()?**

**Answer:** UseRouting() sets up routing middleware to match requests to endpoints. UseEndpoints() maps the matched routes to specific request handlers or controllers.

**402. How do you use IServiceCollection.AddHttpClient() for configuring retry policies?**

**Answer:** Use AddHttpClient() to register HttpClient and configure retry policies using AddTransientHttpErrorPolicy() to handle transient errors.

**403. What is IApplicationBuilder.UseAuthorization() and where should it be used?**

**Answer:** UseAuthorization() adds authorization middleware to the pipeline. It should be used after UseAuthentication() to ensure that authorization occurs after authentication.

**404. How do you implement Custom Middleware` for request logging?**

**Answer:** Create middleware that logs request details in the InvokeAsync method. Register it in Configure using app.UseMiddleware<LoggingMiddleware>().

**405. What is IServiceCollection.AddDbContext() and how is it used with Entity Framework Core?**

**Answer:** AddDbContext() registers DbContext with dependency injection, configuring database connections and options for Entity Framework Core.

**406. How do you configure ASP.NET Core to use Custom Service Providers`?**

**Answer:** Create and configure custom service providers by implementing IServiceProvider or using ServiceProvider for specific service resolutions.

**407. What is IApplicationBuilder.UseExceptionHandler() and how is it used for custom error handling?**

**Answer:** UseExceptionHandler() adds middleware to handle exceptions and redirect to a custom error page or action for displaying error information.

**408. How do you configure ASP.NET Core to use Custom Model Validators`?**

**Answer:** Implement custom model validators by creating classes that inherit from IModelValidator or IModelValidatorProvider. Register them with the service collection.

**409. What is IWebHostBuilder.UseContentRoot() used for in the web host configuration?**

**Answer:** UseContentRoot() sets the application's root directory for locating files and configuring file paths, such as wwwroot or configuration files.

**410. How do you use IApplicationBuilder.UseStaticFiles() with a custom file provider?**

**Answer:** Configure UseStaticFiles() with StaticFileOptions to specify a custom FileProvider for serving static files from a non-default location.

**411. What is IServiceCollection.AddMemoryCache() and when is it appropriate to use?**

**Answer:** AddMemoryCache() registers in-memory caching services, suitable for caching data within the application's memory to improve performance.

**412. How do you implement Custom Authorization Policies` for role-based access control?**

**Answer:** Define custom policies in AddAuthorization() with role requirements. Apply policies to controllers or actions using [Authorize(Policy = "PolicyName")].

**413. What is IApplicationBuilder.UseEndpoints() and how does it interact with UseRouting()?**

**Answer:** UseEndpoints() maps request handlers to routes defined by UseRouting(), enabling endpoint routing for handling requests.

**414. How do you configure ASP.NET Core to use Custom Request Headers`?**

**Answer:** Add middleware to modify request headers in HttpContext.Request.Headers or configure headers in response middleware for custom header management.

**415. What is IServiceCollection.AddDistributedMemoryCache() and when should it be used?**

**Answer:** AddDistributedMemoryCache() registers a distributed memory cache suitable for caching data across multiple instances of the application.

**416. How do you implement Custom Exception Filters` in ASP.NET Core?**

**Answer:** Create custom exception filters by implementing IExceptionFilter or IAsyncExceptionFilter. Register the filters globally or at the controller/action level.

**417. What is IWebHostBuilder.UseStartup() and how is it used?**

**Answer:** UseStartup<TStartup>() specifies the startup class for configuring services and middleware. It sets up the application's request processing pipeline.

**418. How do you configure ASP.NET Core to use Custom Health Check UI`?**

**Answer:** Add health check UI services with AddHealthChecksUI() and configure UI settings. Register the UI middleware to display health check results.

**419. What is IApplicationBuilder.UseDeveloperExceptionPage() and when should it be used?**

**Answer:** UseDeveloperExceptionPage() provides detailed exception information for development. It should be used only in development environments to aid debugging.

**420. How do you implement Custom Logging Providers` in ASP.NET Core?**

**Answer:** Create and configure custom logging providers by implementing ILoggerProvider and registering them in ConfigureLogging for application-wide logging.

**421. What is IApplicationBuilder.UseHttpsRedirection() and how does it enhance security?**

**Answer:** UseHttpsRedirection() automatically redirects HTTP requests to HTTPS, ensuring secure communication between clients and the server.

**422. How do you configure ASP.NET Core to use Custom Routing Constraints`?**

**Answer:** Define custom routing constraints by implementing IRouteConstraint. Register constraints and use them in route templates to enforce routing rules.

**423. What is IServiceCollection.AddOptions() and how is it used for configuration?**

**Answer:** AddOptions() registers services for configuration options. Bind configuration settings to classes and access them via IOptions<T>.

**424. How do you implement Custom Middleware` for response compression?**

**Answer:** Create middleware to compress response content and register it in Configure using app.UseMiddleware<CompressionMiddleware>().

**425. What is IApplicationBuilder.UseStaticFiles() used for, and how do you customize its behavior?**

**Answer:** UseStaticFiles() serves static files from specified directories. Customize behavior using StaticFileOptions to configure file providers and response headers.

**426. How do you configure ASP.NET Core to use Custom Session Stores`?**

**Answer:** Add custom session stores by implementing ISessionStore. Register the custom store with AddSession() and configure it in ConfigureServices.

**427. What is IApplicationBuilder.UseRouting() and how does it support endpoint routing?**

**Answer:** UseRouting() sets up routing middleware that matches requests to endpoints. It enables endpoint routing by defining routing patterns and handlers.

**428. How do you implement Custom Model Binding` for complex scenarios?**

**Answer:** Create a custom model binder by implementing IModelBinder. Register it with ModelBinderProviders to handle complex model binding logic.

**429. What is IWebHostBuilder.UseSetting() used for in the web host configuration?**

**Answer:** UseSetting() sets configuration values for the web host builder, allowing customization of host settings such as URLs and environment variables.

**430. How do you configure ASP.NET Core to use Custom CORS Policies`?**

**Answer:** Add CORS policies using AddCors() in ConfigureServices. Define policies and apply them with UseCors() to control cross-origin resource sharing.

**431. What is IApplicationBuilder.UseAuthentication() and how is it used in the pipeline?**

**Answer:** UseAuthentication() adds authentication middleware to the request pipeline. It should be used before UseAuthorization() to handle user authentication.

**432. How do you implement Custom Health Check Responses` in ASP.NET Core?**

**Answer:** Customize health check responses by implementing IHealthCheck and configuring it with AddHealthChecks() in ConfigureServices. Use HealthCheckResults for custom responses.

**433. What is IServiceCollection.AddDistributedRedisCache() and how is it used?**

**Answer:** AddDistributedRedisCache() registers Redis as a distributed cache provider. Configure Redis settings and use IDistributedCache for cache operations.

**434. How do you use IApplicationBuilder.UseEndpoints() to configure MVC or Razor Pages?**

**Answer:** UseEndpoints() maps MVC controllers or Razor Pages to routes. Define endpoint routing configurations for handling requests and generating responses.

**435. What is IWebHostBuilder.ConfigureAppConfiguration() used for?**

**Answer:** ConfigureAppConfiguration() sets up configuration sources for the application, allowing customization of configuration settings and their sources.

**436. How do you implement Custom Request Middleware` for logging request details?**

**Answer:** Create middleware to log request details in the InvokeAsync method. Register it in Configure using app.UseMiddleware<RequestLoggingMiddleware>().

**437. What is IApplicationBuilder.UseRequestLocalization() used for?**

**Answer:** UseRequestLocalization() adds middleware to localize request responses based on the user's culture or language preferences.

**438. How do you configure ASP.NET Core to use Custom Dependency Injection` scopes?**

**Answer:** Define custom DI scopes using IServiceScopeFactory to create IServiceScope instances. Manage scoped services within the scope context.

**439. What is IServiceCollection.AddMemoryCache() used for, and when is it appropriate?**

**Answer:** AddMemoryCache() registers in-memory caching services for local caching of frequently accessed data, improving application performance.

**440. How do you implement Custom Caching Strategies` with distributed caches?**

**Answer:** Implement caching strategies with distributed caches using IDistributedCache. Configure caching policies and manage cache expiration and eviction.

**441. What is IApplicationBuilder.UseEndpoints() and how does it work with routing?**

**Answer:** UseEndpoints() configures how endpoints are matched and handled based on routes defined by UseRouting(). It completes the routing setup for handling requests.

**442. How do you configure ASP.NET Core to use Custom Static File Providers`?**

**Answer:** Configure custom static file providers using StaticFileOptions and specify a custom FileProvider for serving files from different locations.

**443. What is IServiceCollection.AddControllersWithViews() used for?**

**Answer:** AddControllersWithViews() registers MVC services with support for both controllers and views, suitable for traditional MVC applications with views.

**444. How do you implement Custom Authorization Handlers` in ASP.NET Core?**

**Answer:** Create custom authorization handlers by implementing IAuthorizationHandler. Register handlers with AddAuthorization() to apply authorization policies.

**445. What is IWebHostBuilder.UseStartup() and how is it used for application configuration?**

**Answer:** UseStartup<TStartup>() specifies the startup class for configuring services and middleware in the application. It sets up the application's request pipeline.

**446. How do you configure ASP.NET Core to use Custom Validation Providers`?**

**Answer:** Implement custom validation providers by creating classes that inherit from IValidator<T>. Register providers and use them for model validation.

**447. What is IApplicationBuilder.UseAuthorization() and how does it enforce policies?**

**Answer:** UseAuthorization() adds authorization middleware to the pipeline. It enforces authorization policies configured in AddAuthorization() to secure resources.

**448. How do you configure ASP.NET Core to use Custom Middleware` for handling errors?**

**Answer:** Create custom error-handling middleware to process exceptions and return custom error responses. Register it in Configure using app.UseMiddleware<ErrorHandlingMiddleware>().

**449. What is IServiceCollection.AddDistributedMemoryCache() and how is it different from AddMemoryCache()?**

**Answer:** AddDistributedMemoryCache() registers a distributed in-memory cache for multiple application instances, while AddMemoryCache() registers a local in-memory cache.

**450. How do you implement Custom Model Binding` for complex data structures?**

**Answer:** Create a custom model binder by implementing IModelBinder and register it with ModelBinderProviders for binding complex data types in requests.

Let me know if you need more details or specific areas covered!

4o mini

more 50

ChatGPT

Certainly! Here are another 50 interview questions and answers for .NET Core:

**451. What is IApplicationBuilder.UseRouting() used for in ASP.NET Core?**

**Answer:** UseRouting() sets up routing middleware to route HTTP requests to appropriate endpoints based on route templates.

**452. How do you use IServiceCollection.AddAuthentication() to set up authentication?**

**Answer:** AddAuthentication() adds authentication services to the DI container. Configure schemes and handlers for different authentication methods like JWT, cookies, etc.

**453. What is IApplicationBuilder.UseStaticFiles() and how can you customize its behavior?**

**Answer:** UseStaticFiles() serves static files from the wwwroot folder. Customize behavior using StaticFileOptions to configure file providers, caching, and response headers.

**454. How do you use IServiceCollection.AddCors() to enable CORS in ASP.NET Core?**

**Answer:** AddCors() registers CORS services. Define policies to specify allowed origins, methods, and headers. Use UseCors() to apply these policies.

**455. What is IWebHostBuilder.UseKestrel() and when should it be used?**

**Answer:** UseKestrel() configures the Kestrel server for hosting the application. It's used to set server options like port and URL bindings.

**456. How do you implement Custom Middleware` for request transformation?**

**Answer:** Create middleware to transform requests by modifying HttpContext.Request. Register it in Configure using app.UseMiddleware<RequestTransformationMiddleware>().

**457. What is IServiceCollection.AddDbContext<TContext>() and why is it important?**

**Answer:** AddDbContext<TContext>() registers a DbContext with the DI container, allowing Entity Framework Core to manage database connections and operations.

**458. How do you configure ASP.NET Core to use Custom Error Handling Middleware`?**

**Answer:** Create custom error-handling middleware to catch and handle exceptions. Register it in Configure using app.UseMiddleware<ErrorHandlingMiddleware>().

**459. What is IServiceCollection.AddControllers() and when would you use it?**

**Answer:** AddControllers() registers MVC services for controllers without views. Use it for API-only applications or microservices.

**460. How do you implement Custom Model Binding` for complex data structures?**

**Answer:** Create a custom model binder by implementing IModelBinder. Register it with ModelBinderProviders to handle complex model binding scenarios.

**461. What is IApplicationBuilder.UseAuthentication() and where should it be placed in the pipeline?**

**Answer:** UseAuthentication() adds authentication middleware to the pipeline. Place it before UseAuthorization() to ensure authentication occurs before authorization.

**462. How do you use IServiceCollection.AddMemoryCache() effectively?**

**Answer:** AddMemoryCache() registers in-memory caching services. Use it to cache frequently accessed data within the application's memory for improved performance.

**463. What is IServiceCollection.AddScoped() used for?**

**Answer:** AddScoped() registers a service with a scoped lifetime, creating a new instance per request or connection. It's used for services that need to maintain state within a request.

**464. How do you implement Custom Authorization Filters` for controlling access?**

**Answer:** Create custom authorization filters by implementing IAuthorizationFilter or IAsyncAuthorizationFilter. Register them globally or at the controller/action level.

**465. What is IApplicationBuilder.UseEndpoints() and how does it integrate with routing?**

**Answer:** UseEndpoints() maps endpoint routes to request handlers defined by UseRouting(). It finalizes the routing setup for processing requests.

**466. How do you configure ASP.NET Core to use Custom Data Protection Providers`?**

**Answer:** Add custom data protection providers using AddDataProtection() and configure it for encrypting and protecting sensitive data.

**467. What is IServiceCollection.AddHttpClient() and how is it used for making HTTP requests?**

**Answer:** AddHttpClient() registers HttpClient services, enabling configuration of HTTP clients with settings like base address and policies for making HTTP requests.

**468. How do you use IApplicationBuilder.UseRequestLocalization() for localization?**

**Answer:** UseRequestLocalization() adds localization middleware to handle culture-specific formatting and translations based on user preferences.

**469. What is IServiceCollection.AddOptions() and how is it used for configuration management?**

**Answer:** AddOptions() registers services for configuration options. Bind configuration settings to classes and access them via IOptions<T>.

**470. How do you implement Custom Caching Strategies` with distributed caches?**

**Answer:** Use IDistributedCache to implement caching strategies. Configure cache expiration and eviction policies for distributed caching scenarios.

**471. What is IApplicationBuilder.UseHttpsRedirection() and why is it important?**

**Answer:** UseHttpsRedirection() automatically redirects HTTP requests to HTTPS, ensuring secure communication and protecting data in transit.

**472. How do you configure ASP.NET Core to use Custom Routing Constraints`?**

**Answer:** Define routing constraints by implementing IRouteConstraint. Register constraints and use them in route templates to enforce specific routing rules.

**473. What is IServiceCollection.AddSignalR() and how is it used for real-time communication?**

**Answer:** AddSignalR() registers SignalR services for real-time web communication, enabling features like hubs and persistent connections.

**474. How do you implement Custom Health Checks` in ASP.NET Core?**

**Answer:** Implement custom health checks by creating classes that implement IHealthCheck. Register health checks with AddHealthChecks() and configure endpoints.

**475. What is IApplicationBuilder.UseDeveloperExceptionPage() and when should it be used?**

**Answer:** UseDeveloperExceptionPage() provides detailed exception information for debugging. Use it only in development environments to aid in error diagnosis.

**476. How do you use IServiceCollection.AddDataProtection() for securing data?**

**Answer:** AddDataProtection() registers data protection services for encrypting and securing data, such as cookies and sensitive information.

**477. What is IServiceCollection.AddDistributedRedisCache() and how is it used?**

**Answer:** AddDistributedRedisCache() registers Redis as a distributed cache provider. Configure Redis settings and use IDistributedCache for caching.

**478. How do you implement Custom Request Headers` in ASP.NET Core?**

**Answer:** Add middleware to modify request headers in HttpContext.Request.Headers. Configure headers for custom header management.

**479. What is IWebHostBuilder.ConfigureAppConfiguration() used for?**

**Answer:** ConfigureAppConfiguration() sets up configuration sources for the application, allowing customization of configuration settings.

**480. How do you use IApplicationBuilder.UseAuthorization() to enforce authorization policies?**

**Answer:** UseAuthorization() adds authorization middleware to the pipeline, enforcing policies defined in AddAuthorization() to secure resources.

**481. What is IServiceCollection.AddMvcCore() and how does it differ from AddMvc()?**

**Answer:** AddMvcCore() registers core MVC services without views, while AddMvc() includes full MVC features including views, Razor pages, and more.

**482. How do you configure ASP.NET Core to use Custom Request Localization`?**

**Answer:** Add localization services and configure RequestLocalizationOptions to support localization based on user preferences and culture settings.

**483. What is IApplicationBuilder.UseRouting() and how does it support endpoint routing?**

**Answer:** UseRouting() sets up routing middleware to match requests to endpoints based on route templates, enabling endpoint routing.

**484. How do you implement Custom Exception Filters` for handling exceptions in ASP.NET Core?**

**Answer:** Create custom exception filters by implementing IExceptionFilter or IAsyncExceptionFilter. Register them globally or at the controller/action level.

**485. What is IServiceCollection.AddHealthChecks() used for?**

**Answer:** AddHealthChecks() registers health check services, allowing you to define and monitor the health status of application components.

**486. How do you configure ASP.NET Core to use Custom Session Providers`?**

**Answer:** Add custom session providers by implementing ISessionStore. Register the custom store with AddSession() and configure it in ConfigureServices.

**487. What is IApplicationBuilder.UseEndpoints() and how does it work with routing?**

**Answer:** UseEndpoints() maps endpoint routes to request handlers defined by UseRouting(). It finalizes the routing setup for handling requests.

**488. How do you implement Custom Request Middleware` for logging requests?**

**Answer:** Create middleware that logs request details in the InvokeAsync method. Register it in Configure using app.UseMiddleware<RequestLoggingMiddleware>().

**489. What is IWebHostBuilder.UseContentRoot() used for?**

**Answer:** UseContentRoot() sets the root directory for the application, determining where files and configuration settings are located.

**490. How do you use IServiceCollection.AddHttpClient() for configuring retry policies?**

**Answer:** Use AddHttpClient() to register HttpClient and configure retry policies using AddTransientHttpErrorPolicy() to handle transient errors.

**491. What is IApplicationBuilder.UseStaticFiles() and how can you customize its behavior?**

**Answer:** UseStaticFiles() serves static files from the wwwroot folder. Customize its behavior with StaticFileOptions for file providers and caching.

**492. How do you configure ASP.NET Core to use Custom Logging Providers`?**

**Answer:** Add custom logging providers by implementing ILoggerProvider and registering them with AddLogging() to control logging behavior.

**493. What is IServiceCollection.AddDbContextPool<TContext>() used for?**

**Answer:** AddDbContextPool<TContext>() registers a pooled DbContext to improve performance by reusing DbContext instances, reducing overhead.

**494. How do you implement Custom Authorization Middleware` for complex authorization logic?**

**Answer:** Create custom authorization middleware by implementing IMiddleware or IAsyncMiddleware. Register and configure it to enforce complex authorization rules.

**495. What is IApplicationBuilder.UseHttpsRedirection() and when should it be used?**

**Answer:** UseHttpsRedirection() automatically redirects HTTP requests to HTTPS. Use it to ensure secure communication and protect data in transit.

**496. How do you configure ASP.NET Core to use Custom Model Validation`?**

**Answer:** Implement custom model validators by creating classes that inherit from IModelValidator. Register them with AddMvc() for custom validation logic.

**497. What is IServiceCollection.AddMemoryCache() used for, and how is it different from AddDistributedMemoryCache()?**

**Answer:** AddMemoryCache() registers in-memory caching for a single application instance, while AddDistributedMemoryCache() is used for distributed caching across multiple instances.

**498. How do you implement Custom Exception Handling` in ASP.NET Core?**

**Answer:** Create custom exception-handling middleware to catch and handle exceptions. Register it in Configure using app.UseMiddleware<ExceptionHandlingMiddleware>().

**499. What is IApplicationBuilder.UseRouting() and how does it fit into the request pipeline?**

**Answer:** UseRouting() sets up routing middleware, allowing requests to be routed to appropriate endpoints based on route templates. It should be placed before UseEndpoints().

**500. How do you use IServiceCollection.AddIdentity<TUser, TRole>() for identity management?**

**Answer:** AddIdentity<TUser, TRole>() registers services for managing user identities and roles, including authentication and authorization features.

**Filter the Duplicate Questions in this file**[**https://www.youtube.com/watch?v=Dl02Rc000t4**](https://www.youtube.com/watch?v=Dl02Rc000t4) **Design like this tempalte**