Final Test

Devanathan-106807

1. What is Delegate?

A delegate is a type that represents **references to methods** with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. You can invoke (or call) the method through the delegate instance.

Delegates are used to pass methods as arguments to other methods.

2. Difference between Delegate and Events?

- Delegate is an object used as a function pointer to hold the reference of a method. On the other hand, events provide an abstraction to delegates.
- A keyword required to declare a delegate is a delegate whereas, a keyword required to declare an event is event.
- A delegate is declared outside a class whereas, an event is declared inside a class.
- To invoke a method using a delegate object, the method has to be referred to the delegate object. On the other hand, to invoke a method using an event object the method has to be referred to the event object.
- Covariance and Contravariance provide extra flexibility to the delegate objects. On the other hand, event has no such concepts.
- Event Accessor handles the list of event handlers whereas delegate has no such concept.
- Delegates are independent on events but, events can not be created without delegate.
- A delegate can be passed as a method parameter. An event is raised but cannot be passed as a method parameter.
- Delegate includes Combine() and Remove() methods to add methods to the invocation list. EventInfo class inspect events and to hook up event handlers that include methods AddEventHandler() and RemoveEventHandler() methods to add and remove methods to invocation list, respectively.
- Delegates are useful as they support events, and they provide the facility of executing the method at runtime. The event accessor allows us to synchronize the event handlers in multithreading applications.

3. What are the differences between a Thread and a Task ?

Thread:

- Threads are units of execution within a process.
- They are managed by the operating system.
- They have a higher overhead compared to tasks.
- They can be foreground or background.
-

Task:

- Tasks are a higher-level abstraction for asynchronous programming.
- They are managed by the Task Parallel Library (TPL) in .NET.
- They have lower overhead and are more lightweight than threads.
- They can represent both synchronous and asynchronous operations.

4. When will you use a Task ?

- Performing time-consuming operations asynchronously.
- Avoiding blocking the main thread, ensuring application responsiveness.
- Enabling parallel execution of multiple tasks.
- Enhancing the performance of I/O-bound operations.
- Improving the user experience in graphical user interface (GUI) applications.
- Handling multiple operations concurrently for increased efficiency.
- Facilitating smoother execution of background processes.

5. Differences between HTTP and TCP

HTTP (Hypertext Transfer Protocol):

- It is a higher-level application layer protocol used for transmitting data over the internet.
- It operates on top of TCP.
- It is stateless, meaning each request-response cycle is independent.

TCP (Transmission Control Protocol):

- It is a lower-level transport layer protocol responsible for establishing and maintaining a reliable connection between two nodes on a network.
- It is connection-oriented, meaning it ensures data is reliably delivered.

6. What is a Web API?

A Web API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other over the internet. In the context of web development, a Web API typically refers to an API that can be accessed over the HTTP protocol.

7. what are the differences between a MVC Application and Web API?

MVC Application:

- MVC (Model-View-Controller) is a design pattern for building interactive and maintainable web applications.
- It is primarily used for building user interfaces.

Web API:

- Web API is used for building APIs that can be consumed by clients like web browsers, mobile apps, etc.
- It focuses on providing data and functionality, not on rendering views.

8. What is DI ?

Dependency Injection (DI) is a design pattern used in software development to achieve Inversion of Control (IoC) and promote loose coupling between components within an application. It involves passing dependencies (objects or services) as parameters to a component (usually a class or method) rather than having the component create or manage them internally.

Here are the key concepts of Dependency Injection:

Inversion of Control (IoC):
- In traditional programming, the flow of control is determined by the program's logic. In IoC, the control flow is inverted because a framework or container manages the flow and decides when to call certain methods or components.

Components:
- Components refer to the various parts or units of an application, such as classes or modules.

Dependency:
- A dependency is an object or service that another component relies on to perform its tasks. For example, a class that sends emails may depend on an email sending service.

Container/Injector:
- The container is a framework or mechanism responsible for managing the dependencies and providing them to the dependent components. It resolves dependencies and injects them into the components that need them.

Types of Dependency Injection:
- **Constructor Injection**: Dependencies are passed through the constructor of a class. This is the most common type of DI and ensures that the dependency is available as soon as the object is created.
- **Method/Setter Injection**: Dependencies are provided through setter methods or properties after the object is constructed. This allows for optional or changeable dependencies.
- **Interface Injection**: The dependent class implements an interface that provides a method for setting the dependency.

Benefits:
- **Decoupling**: Components are not tightly coupled to specific implementations of their dependencies, making them more modular and easier to maintain.
- **Testability**: Dependencies can be easily mocked or replaced during unit testing, allowing for more comprehensive and isolated testing.
- **Flexibility**: It enables the application to be more adaptable to changes, as new implementations of dependencies can be easily introduced.
- **Reusability**: Dependencies can be shared across multiple components, promoting code reuse.

In summary, Dependency Injection is a powerful design pattern that promotes loose coupling, testability, and maintainability in software applications. It enables more modular and adaptable code by allowing dependencies to be provided externally, rather than being created internally within a component.

9. What do you mean by a Middleware?

Middleware is a key concept in web development, particularly in frameworks and platforms like ASP.NET, Express.js, and Django. It refers to a series of functions or components that are executed in the request-response cycle of a web application. These functions have access to the request object (HTTP request) and response object (HTTP response) and can perform various operations, such as modifying the request, processing data, or altering the response.

10. What is the role of an ORM ?

An Object-Relational Mapping (ORM) serves as a bridge between an application's object-oriented code and a relational database. Its primary role is to facilitate the interaction between the application's programming language (which typically uses objects) and the database (which stores data in tables and rows).

11. What are the 2 Data migration options in EFCore. Give the commands for both.

Code-First Migrations:

Code-First Migrations is a technique in EF Core that allows developers to manage database schema changes through code. It involves creating migration files that describe the changes to be made to the database. These migration files are generated based on the changes made in the code.

Add Migration:

To create a new migration file, open the command prompt or terminal and navigate to your project directory. Then, run the following command: "**dotnet ef migrations add <MigrationName>"**
Apply Migrations:After creating a migration, you need to apply it to the database. Use the following command:
dotnet ef database update

- **Revert Migrations**:If needed, you can revert a migration using the following command:

phpCopy code
dotnet ef database update <PreviousMigrationName>

Database-First Approach:

- The Database-First Approach involves reverse engineering an existing database to create EF Core models and context. This approach is useful when working with an existing database and you want to generate the corresponding code.

Command for Database-First Approach:

- **Scaffold DbContext**:To scaffold a DbContext from an existing database, open the command prompt or terminal and run the following command, replacing **<Connection String>** and **<Provider>** with your specific values:

dotnet ef dbcontext scaffold "<Connection String>"Microsoft.EntityFrameworkCore.<Provider> -o Models

12. What is the purpose of a repository ?

    A repository is a design pattern that acts as a mediator between the application's business logic and the data source. It provides an abstraction layer for data access, allowing the application to work with objects instead of directly interacting with the database. This promotes separation of concerns and improves maintainability.

13. What are the benefits of having the connection string in a configuration file ?

    - Separation of concerns: It separates sensitive information (like connection strings) from the application code, making it easier to manage configurations.
    - Security: It reduces the risk of accidentally exposing sensitive information in code repositories.
    - Flexibility: It allows for easy configuration changes without modifying the code, which is especially useful in different deployment environments.

14. What are ViewModel Classes. What purpose do they serve in a N'Tier Application ?

    ViewModel classes are used to shape and represent the data that is specifically needed by the user interface. They serve as an intermediary between the business logic and the presentation layer, providing exactly the data required for a particular view or page.

15. What is a DBContext ?

    DbContext is a class in Entity Framework that represents the session with the database. It is responsible for managing the connection, tracking changes to entities, and performing database operations.

16. What are the various states of an Entity.

- **Detached**: The entity is not being tracked by the **DbContext**.
- **Unchanged**: The entity is being tracked, and its state has not changed since it was queried or attached.
- **Modified**: The entity is being tracked, and its properties have been modified.
- **Added**: The entity is being tracked, and it represents a new record to be inserted into the database.
- **Deleted**: The entity is being tracked, and it represents a record to be deleted from the database.

17. List 4 HTTP Methods used in RESTful applications. Explain when to use them.?

- **GET**: Used to retrieve information from the server.
- **POST**: Used to submit data to be processed by the server (e.g., form submissions).
- **PUT**: Used to update an existing resource with new data.
- **DELETE**: Used to remove a resource from the server.

18. What is the use of DataAnnotation ?

    Data Annotations are attributes in .NET that can be applied to model properties to specify constraints, validations, and other metadata about the data. They are used to control how data is displayed, validated, or persisted in the database.

19. Explain OutPutCaching ?

    Output Caching is a technique used to store the output of a web page or API call in memory or on disk, so that subsequent requests for the same resource can be served directly from the cache without re-executing the entire request.

20. Why are JSON files preferred instead of XML files.?

    JSON (JavaScript Object Notation):

    - Syntax: Based on JavaScript object literals, uses key-value pairs.
    - Readability: Compact, considered more readable and less verbose.
    - Usage: Common in web-based applications and APIs, especially in JavaScript environments.
    - Parsing: Easier and faster to parse in JavaScript environments.
    - Schema and Validation: Supports JSON Schema for data validation.

    XML (eXtensible Markup Language):

    - Syntax: Uses tags with opening and closing elements, forms a tree-like structure.
    - Readability: Can be more verbose, especially for complex nested structures.

- Usage: Historically used for configuration files, document storage, and data interchange.
- Parsing: Requires more effort and resources to parse, involves a more complex process.
- Schema and Validation: Supports Document Type Definitions (DTD) and XML Schema Definition (XSD).
-

21. What is the purpose of Nuget Package Manager?

    NuGet is a package manager for .NET that simplifies the process of adding, removing, and managing third-party libraries or packages in a project. It allows developers to easily integrate existing code and libraries into their projects, reducing the need to reinvent the wheel.

22. In LINQ why should we use FirstOrDefault() method ?

    FirstOrDefault() is used in LINQ to retrieve the first element from a collection that matches a specified condition, or the default value if no matching element is found. This is useful when you want to retrieve a single element from a collection and handle the case where no matching element exists.

23. Write the LINQ code to fetch a Customer by his ID and fetch all the Orders placed by the customer ?

    ```
    var customer = dbContext.Customers.FirstOrDefault(c => c.ID == customerId);

    var orders = dbContext.Orders.Where(o => o.CustomerID == customerId).ToList();
    ```

24. How does DBContext create the sql Query for the Attached entity Objects before saving them?

    Entity Framework's DbContext uses a process called Change Tracking to monitor changes made to attached entity objects. When you call **SaveChanges()**, EF Core generates SQL queries based on the changes tracked in the DbContext and executes them against the database to persist the changes.

25. What are Transaction Boundaries?

    Transaction boundaries define the scope within which a series of operations are treated as a single unit. In databases, transactions ensure that a series of operations either all succeed or fail together, maintaining data integrity.

26. Explain Version Control System?

    A VCS is a system that records changes to files over time, allowing multiple contributors to collaborate on a project. It tracks changes, maintains versions, and enables easy collaboration and coordination among team members.

27. Differences between Local repository and Remote repository

- **Local Repository**:
  A local repository is a copy of a project's files and version history that is stored on a developer's local machine. It allows the developer to work on the project offline and make changes without impacting the shared codebase.

- **Remote Repository**:
  A remote repository is a version-controlled copy of a project that is hosted on a remote server or platform (like GitHub, GitLab, etc.). It serves as a centralized location for collaboration, where multiple developers can push and pull changes.

28. What are the differnece between Pull and Fetch git commands ?

- **Pull**: Pull combines the process of fetching changes from a remote repository and merging them into the current branch in one step. It updates the local branch with the latest changes from the remote.
- **Fetch**: Fetch only retrieves the latest changes from the remote repository and stores them in the local repository. It doesn't automatically merge them into the working branch. It's useful for reviewing changes before merging.

29. What is Staging in git?

Staging in Git refers to the process of preparing files for a commit. Files that are staged are marked to be included in the next commit. This allows you to selectively include specific changes while leaving others out of the commit.

- git add .
- git status
- git commit -m “xyz”
- git pull

30. What is a PULL request?

A Pull Request (PR) is a feature in Git-based version control systems that allows developers to propose changes to a project's codebase. It serves as a request to review and merge the proposed changes from one branch into another, typically from a feature branch into the main or master branch. It facilitates collaboration and code review in a controlled manner.