**Data Access and REST API**

Sample Questions Only

© FDM Group Ltd 2022. All Rights Reserved.

Any unauthorised reproduction or distribution in part  
or in whole will constitute an infringement of copyright.

# Overview

This document shows 26 sample questions that can be used for knowledge assessment for trainees that have completed the modules on introduction to Data Access and REST API.

1. What is a database? Which one are you using for your project and why.

Database is a structured collection of data that is stored and managed in a way that allows for efficient retrieval, updating, and manipulating of that data

I am using SQL Server

* + Ease of use: SQL Server tools, such as SSMS simplifies database administration and development tasks

1. What is Entity Framework?

EF is an object relational mapper (ORM) framework developed by Microsoft for .NET applications. It simplifies database access and manipulation using object-oriented code rather than writing raw SQL queries.

1. List the four different approaches (known as workflows) Entity Framework offers.

Which one are you using for your project and why.

* + Database First
  + Model First
  + Code First (to a new database)
  + Code First (to an existing database)

I used Code First (to a new database) the main reason for this choose was that I wanted to create a new database and its schema based on the entity classes and data model defined in the code.

1. Describe Entity Framework Database First Modelling and where you have used it and why.

Database First approach is when you have an existing database schema and want to generate entity classes and a data model from the database.

I used this approach in the REST API project and the reason for the choose was because I had an existing database and wanted to leverage EF for data access. Database First allowed me to generate the model from the database.

1. Describe Repository design pattern and where you have used it and why.

The Repository design pattern is an architectural pattern in software development, used when dealing with data access. It provides an abstraction layer between the business logic and the data source. It hides the details of how data is retrieved, stored, and manipulated – making it easier to change the data source or storage technology.

I used the repository pattern in ADO.NET project and ASP.NET Web API project to create a clean separation between the application’s business logic and the data access code. It also helped to provide an abstraction layer over data access, allowing me to work with data entities using a consistent API, regardless of the actual data source.

1. Describe Data Entity Classes and where you have used it and why.

Classes that model the data structures so that we can manipulate them effectively in our program. The classes do not inherit features from any framework specific base class or interface.

I used data entity classes in my Entity Framework Plain CLR Object (POCO) classes to create the class-to-database mapping. The reason I used Data Entity classes was to create a direct one-to-one mapping between the classes and the database tables.

1. Describe DbContext Class and where you have used it and why.

The DbContext is a component of EF and it is a lightweight and cross-platform version of EF, and it serves as the bridge between .NET applications and databases.

I have used the DbContext class in my EF Poco project for the following reasons:

* Data Operations: The DbContext provides methods for performing various data operations, including getting data, inserting new records, updating existing records, and deleting records.
* Database Initialisation and Configuration: The DbContext include configuration options that allow to define relationship, and other aspects of the database schema.
* Communication Between Application and Database: The DbContext class acts as an intermediary that facilitated communication between the application and the database.

1. Describe the acronym CRUD and where you have used it and why.

CRUD stands for CREATE, READ, UPDATE, and DELETE. It represents the four basic operations for managing data in a database.

* + Create
    - Involves adding new records to a database
  + Read
    - Involves retrieving existing records from a database
  + Update
    - Involves modifying existing records in a database
  + Delete
    - Involves removing records from a database

I used the CRUD operation in all my projects, and the reason for the usage was for data management. They enable the creation, retrieval, modification, and removal of data from the database.

1. In a **.NET Framework** project that uses Entity Framework, where would you put the configuration information for the application?

The App.config file

1. In a .NET Framework project that uses Entity Framework, describe one key configuration information that you must declare for the application?

One key configuration information that you must declare for the application is the database Connection string. The connection string specifies the details required to establish a connection to the underlying database.

1. In a **.NET Core** project that uses Entity Framework, where would you put the configuration information for the application?

appSetting.json file

1. When implementing security, what do you understand by the terms Authentication, Authorization, and Accounting (AAA). Which one are you using in your project and why.

* Authentication: is the process of verifying the identity of a user. Help ensure that only authorise user can access the system, safeguarding sensitive data and resources from unauthorised access.
* Authorization: is the process of determining whether a user has access to a resource. Ensure that authenticated users only have the access to the resources and actions they are allowed to use.
* Accounting (Auditing): tracking and recording activities taken within the application for monitoring and accountability purposes. Help when you need to log and track user actions within the application

1. Describe ASP.NET Core Identity Management.

ASP.NET Core Identity is an identity framework built into ASP.NET Core, it is designed to simplify user authentication, authorization, and user data management in web applications.

* User Authentication

Identify handles user authentication by providing built-in mechanism for username and password-based login.

* User Authorisation and Roles

Supports role-based authorization, allows for defining roles and assigning them to users.

* User Registrations

Simplifies user registration, including email confirmation and password reset functionalities.

* Token-based Authentication

Works seamlessly with token-based authentication, allowing you to integrate it with RESTful APIs.

1. When implementing security, what do you understand by the term JSON Web Token (JWT). Where have you used it and why.

JSON Web Token (JWT) is a token used for securely transmitting information between parties.

I have used JWT in my ASP.NET Core Web API project for authentication and authorization of user in the following context:

* Users receive a token upon successful login. The JWTs are used to verify the identity of users who want to access protected resources in my web application.

1. In a ASP.NET Core REST API application, what is the difference between a Controller class and a Repository class?

* Controller class

The Controller class is responsible for handling incoming HTTP requests, processing them, and generating appropriate HTTP responses. Act as an interface between the client and the underlying application logic and data.

A controller will have methods to handle HTTP requests and interaction with the Repository to retrieve or manipulate data.

* Repository class

The Repository class is responsible for data access and interaction with the database. It encapsulates the logic required to perform CRUD operation of the data

The Repository methods interact directly with the database to retrieve or modify the records.

The separation of concerns between Controller and Repositories follows good software design principles, such as the Single Responsibility Principle, which encourages modularity and maintainability in the application. Controllers focus on handling HTTP-related concerns, while Repositories focus on data access and database interactions.

1. What is Cross Origin Resource Sharing (CORS)?

CORS is a security feature implemented in the web browser that allows or restricts web applications running at one origin to make requests for resources from a different origin.  
  
It defines a set of rules and HTTP headers that enable web servers to specify which origin are permitted to access their resource. It prevents potentially malicious web applications from making unauthorized cross-origin requests for sensitive data.

1. In a ASP.NET Core REST API application, how would you configure your application for Cross Origin Resource Sharing (CORS)?

By specifying which origins are allowed to access API resource.  
  
Steps:

* Install the CORS package
* Configure the Program.cs file
* Add the CORS middleware using UseCors()
* Enable Core with [EnableCors] attribute, and apply a named policy to only the endpoints that require CORS
* Use [EnableCors(“MyPolicy”)] to enable the “MyPolicy” CORS policy for the controller/endpoint
* Use [DisableCors] to disable CORS for a method/endpoint

1. When building, testing and consuming RESTful web services, what is the difference between swagger and Postman?

|  |  |
| --- | --- |
| Swagger | Postman |
| Used for documenting RESTful APIs. | Used for testing and interacting with RESTful APIs. |
| Provides interactive API documentation (Swagger UI) | Provides a user-friendly interface for creating and sending HTTP request with different methods (GET, POST,PUT, DELETE, etc.) |

1. What is an API?

API stands for Application Programming Interface – is a set of rules, protocols, and tools that allows different software applications to communicate with each other.

Key characteristics:

* Security: APIs can implement security mechanism such as authentication and authorization to control access to their functions
* Interoperability: APIs enable different software components to communicate and exchange data.

1. Describe RESTful web services

RESTful web services, also referred to as REST API, are a type of web service that follows the principles and constraint of REST architecture. RESTful Web services allow web applications to communicate with each other.

Characteristics of RESTful web services:

* Stateless: Each HTTP request from a client to a server must contain all the information needed to understand and process the request. The server should not store client state.
* Resource-Based: In REST, everything is treated as a resource, which can be an object, data, or service. Resources are identified by unique URIs, and the representation of a resource can be in various formats, such as JSON, XML.
* Client-Server: The client and server are separate entities that communicate over a stateless connection. The client is responsible for the user interface and user experience, while the server is responsible for processing requests and managing resources.

1. What is Scaffold-DbContext command use for? Where have you used it and why.

The Scaffold-DbContext command is a feature provided by EF Core, and it is used to generate models class and a DbContext class from an existing database schema. It automates the process of creating entity classes that map to the database tables and the associated DbContext for data access.

I used the Scaffold-DbContext command in my REST API project where I used EF database first workflow to create entity classes that map to database tables and the associated DbContext for data access.

The Reason I used Scaffold-DbContext:

* Timesaving: Manually creating entity classes and configuration relationships can be a time-consuming task. Scaffold-DbContext automates this process, saving development time.
* Maintaining Consistency: It ensures that the code models match the database scheme accurately. Any changes to the database schema can be quickly reflected in the code by re-running Scoffolold-DbContext

1. What is Dependency Injection? Where have you used it and why.

Dependency injection

* Separation of Concerns: DI help separate concerns within your code. It allows you to isolate the behaviour of individual components and make your code more modular
* Flexibility and Reusability: DI enables you to change implementation or dependencies without modifying the code of the dependent class. This allows for code you to reuse components in different contexts.
* Loose coupling: DI leads to loose coupling between classes, reducing the dependencies and making it easier to replace or extend components without affecting the entire application.

1. In a ASP.NET Core REST API application, what do you understand by the term end point?

Endpoints refer to a specific URL (Uniform Resource Locator) or (Uniform Resource Identifier) that represents a resource or a specific operation within the API. Endpoints define the API’s functionality and the operations that clients can perform.

1. In a ASP.NET Core REST API application, how would you configure an end point for logged in users only?

In a ASP.NET Core REST API application, you can configure an endpoint to be accessible only to logged in users by applying authentication and authorization middleware.

Steps:

1. Add authentication middleware to the Program.cs file. Use a suitable authentication scheme like JWT, cookies, or OAuth, depending on your application’s requirements
2. Call the app.UseAuthentication() and app.UseAuthorization() methods. The order of these middleware is essential.
3. Authorize Endpoints: To restrict access to an endpoint for logged-in users you can use the [Authorize] attribute on the controller action that corresponds to the endpoint. The attribute ensures that only authorized users can access the endpoint.
4. In a ASP.NET Core REST API application, how would you configure an end point for all users with a specific role?

In a ASP.NET Core REST API application, you can configure endpoints for users with a specific role by following the steps below

1. Add Authentication and Authorization middleware
2. Configure identity by using the IdentityRole class to represent roles. The Identity Role class will help you in defining roles and storing user and role information in the database.
3. Create a Helper method – to check if a user belongs to a specific role.
4. In the API controller or endpoint, use the helper method to check if a the authenticate user belong to a specific role
5. Use the [Authorize] attribute to restrict access to users who are not authorized
6. Describe Microservices? What is the difference between Microservices and REST API?

|  |  |
| --- | --- |
| Microservices | REST AP |
| Is an architectural style and approach to designing and organising an application into small, independent services | Is an architectural style that define a set of constraint for designing applications. REST is concerned with the design of HTTP-based APIs for communication between the client and servers. |
| Communicate with each through well-defined protocols, which can include RESTful HTTP APIs but are not limited to them | REST APIs are a specific approach to designing HTTP-based communication between clients and servers. |