**Basic Web API Concepts**

**1. What is a Web API? How is it different from a web application?**

A **Web API** (Application Programming Interface) is a **service that allows communication between applications over HTTP**. It provides endpoints to send and receive data in formats like JSON or XML.

* **Web Application**: Designed for user interactions through a UI.
* **Web API**: Designed for machine-to-machine communication, without a UI.

**2. What is the difference between Web API and MVC?**

| **Feature** | **Web API** | **MVC** |
| --- | --- | --- |
| Purpose | Exposes data/services over HTTP | Handles user interface and business logic |
| Response Format | JSON, XML | HTML, JSON |
| HTTP Methods | GET, POST, PUT, DELETE | Typically GET and POST |

**3. What are the advantages of using Web API?**

* Supports **RESTful architecture**.
* Works with **multiple clients** (mobile apps, web apps, etc.).
* Uses **lightweight formats** like JSON/XML.
* **Platform-independent**.

**4. What are HTTP methods? Explain GET, POST, PUT, DELETE with examples.**

* **GET**: Retrieves data. Example: GET /api/products
* **POST**: Creates new data. Example: POST /api/products
* **PUT**: Updates existing data. Example: PUT /api/products/1
* **DELETE**: Removes data. Example: DELETE /api/products/1

**5. What is RESTful API? What are its principles?**

A **RESTful API** follows REST (**Representational State Transfer**) principles:

1. **Stateless** – Each request is independent.
2. **Client-Server** – Separation between UI and data storage.
3. **Cacheable** – Responses can be cached.
4. **Layered System** – Can have security and logging layers.
5. **Uniform Interface** – Uses standard HTTP methods.

**6. What is the difference between SOAP and REST?**

| **Feature** | **SOAP** | **REST** |
| --- | --- | --- |
| Protocol | Uses XML-based protocol | Uses HTTP with JSON/XML |
| Performance | Slower | Faster |
| Complexity | More complex | Simpler |
| State | Stateful | Stateless |

**7. What is the difference between Web API 2 and .NET Core Web API?**

* **Web API 2** (older) runs on **.NET Framework**, supports only Windows.
* **.NET Core Web API** is cross-platform, faster, and lightweight.

**8. What is an HTTP status code? Name some common status codes.**

HTTP status codes indicate the result of an HTTP request:

* **200 OK** – Request successful.
* **201 Created** – New resource created.
* **400 Bad Request** – Invalid request format.
* **401 Unauthorized** – Authentication required.
* **404 Not Found** – Resource not found.
* **500 Internal Server Error** – Server-side issue.

**9. What is the role of [Route] and [HttpGet], [HttpPost] in Web API?**

* **[Route]**: Defines URL pattern. Example: [Route("api/products")]
* **[HttpGet]**: Marks a method for GET requests. Example:

[HttpGet]

public IActionResult GetProducts() { return Ok(products); }

* **[HttpPost]**: Handles POST requests. Example:

[HttpPost]

public IActionResult AddProduct(Product p) { /\* insert logic \*/ }

**10. What is the difference between IActionResult and ActionResult in ASP.NET Core Web API?**

* **IActionResult**: More flexible, can return different types like Ok(), NotFound(), BadRequest().
* **ActionResult<T>**: Returns a specific data type. Example: ActionResult<Product>.

**Entity Framework & Database Handling**

**11. What is Entity Framework (EF) Core?**

EF Core is an **ORM (Object-Relational Mapper)** that allows developers to work with databases using **C# objects** instead of SQL queries.

**12. What is the difference between Code First and Database First approaches?**

* **Code First**: Define models in C#, then generate the database.
* **Database First**: Start with a database, then generate C# models.

**13. How do you use EF Core in Web API?**

1. Install EF Core package.
2. Create a **DbContext** class.
3. Configure connection string in appsettings.json.
4. Use DbSet<T> for tables.

**14. What is Dependency Injection, and how is it used in .NET Core Web API?**

Dependency Injection (DI) is a design pattern where dependencies are **injected into a class** instead of being created inside it.

Example:

services.AddDbContext<AppDbContext>(options => options.UseSqlServer("connectionString"));

**15. How can you perform CRUD operations in Web API using EF Core?**

* **Create**: dbContext.Products.Add(product);
* **Read**: var product = dbContext.Products.Find(id);
* **Update**: dbContext.Products.Update(product);
* **Delete**: dbContext.Products.Remove(product);

**Middleware & Authentication**

**16. What is middleware in .NET Core?**

Middleware is a component in the request pipeline that handles **logging, authentication, CORS, etc.**

**17. What is CORS (Cross-Origin Resource Sharing), and why is it needed in Web API?**

CORS allows **requests from different domains** to access the API.

**18. How do you enable CORS in ASP.NET Core Web API?**

Add this in Program.cs:

app.UseCors(policy => policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

**19. What are authentication and authorization in Web API?**

* **Authentication**: Confirms identity (login).
* **Authorization**: Grants permissions (access control).

**20. What is JWT (JSON Web Token), and how is it used for authentication?**

JWT is a **token-based authentication** mechanism used in Web APIs. The token is sent in the **Authorization header**.

**Advanced & Miscellaneous**

**21. What is Swagger, and why is it used in Web API?**

Swagger generates **API documentation** and allows testing of endpoints.

To enable Swagger in ASP.NET Core:

app.UseSwagger();

app.UseSwaggerUI();

**22. What is Model Binding in Web API?**

Model binding converts **incoming HTTP requests** into C# objects automatically.

**23. What is Model Validation? How do you perform validation in Web API?**

Model validation checks input data using attributes like:

public class Product {

[Required] public string Name { get; set; }

[Range(1, 100)] public int Price { get; set; }

}

**24. What is the difference between Asynchronous (async/await) and Synchronous API calls?**

* **Synchronous**: Blocks execution until task completes.
* **Asynchronous**: Uses async/await to avoid blocking.

**25. How can you handle exceptions in Web API?**

Use **try-catch** or UseExceptionHandler().

Example:

app.UseExceptionHandler("/error");