Entity Framework

EF Core is an object-relational mapper (ORM). Object-relational mapping is a technique that enables developers to work with data in object-oriented way by performing the work required to map between objects defined in an application's programming language and data stored in relational datasources.

It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with many databases, including SQL Database (on-premises and Azure), SQLite, MySQL, PostgreSQL, and Azure Cosmos DB.

Entity Framework Core (EF Core) is the latest version of the Entity Framework from Microsoft. It has been designed to be lightweight, extensible and to support cross platform development as part of Microsoft's .NET Core framework. It has also been designed to be simpler to use, and to offer performance improvements over previous versions of Entity Framework.

DbContext

The DbContext class is an integral part of Entity Framework. An instance of DbContext represents a session with the database which can be used to query and save instances of your entities to a database. DbContext is a combination of the Unit Of Work and Repository patterns.

To use DbContext in our application, we need to create the class that derives from DbContext, also known as context class. This context class typically includes DbSet<TEntity> properties for each entity in the model.

DbSet

The DbSet class represents an entity set that can be used for create, read, update, and delete operations.

The context class (derived from DbContext) must include the DbSet type properties for the entities which map to database tables and views.

LINQ

LINQ (Language Integrated Query) is uniform query syntax in C# and VB.NET to retrieve data from different sources and formats. It is integrated in C# or VB, thereby eliminating the mismatch between programming languages and databases, as well as providing a single querying interface for different types of data sources.

LINQ is a structured query syntax built in C# and VB.NET to retrieve data from different types of data sources such as collections, ADO.Net DataSet, XML Docs, web service and MS SQL Server and other databases.

Lambda expression

You use a lambda expression to create an anonymous function. Use the lambda declaration operator => to separate the lambda's parameter list from its body. A lambda expression can be of any of the following two forms:

- Expression lambda that has an expression as its body:

(input-parameters) => expression

- Statement lambda that has a statement block as its body:

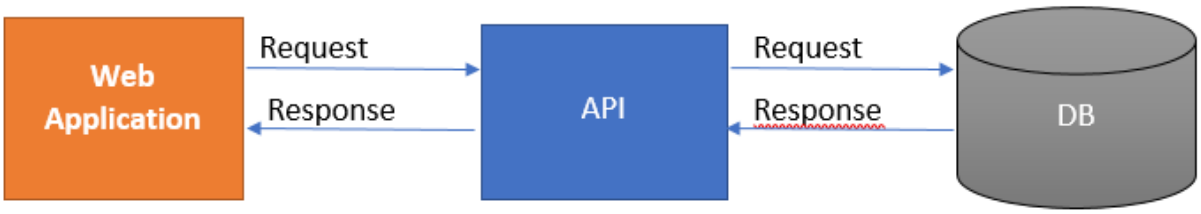
(input-parameters) => { <sequence-of-statements> }

To create a lambda expression, you specify input parameters (if any) on the left side of the lambda operator and an expression or a statement block on the other side.

Any lambda expression can be converted to a delegate type. The delegate type to which a lambda expression can be converted is defined by the types of its parameters and return value. If a lambda expression doesn't return a value, it can be converted to one of the Action delegate types; otherwise, it can be converted to one of the Func delegate types.

Web API

API stands for Application Programming Interface. It is an intermediate software agent that allows two or more applications to interact with each other.



ASP.NET Core supports creating RESTful services, also known as web APIs, using C#. To handle requests, a web API uses controllers. Controllers in a web API are classes that derive from ControllerBase. This article shows how to use controllers for handling web API requests.

A web API consists of one or more controller classes that derive from ControllerBase. The web API project template provides a starter controller:

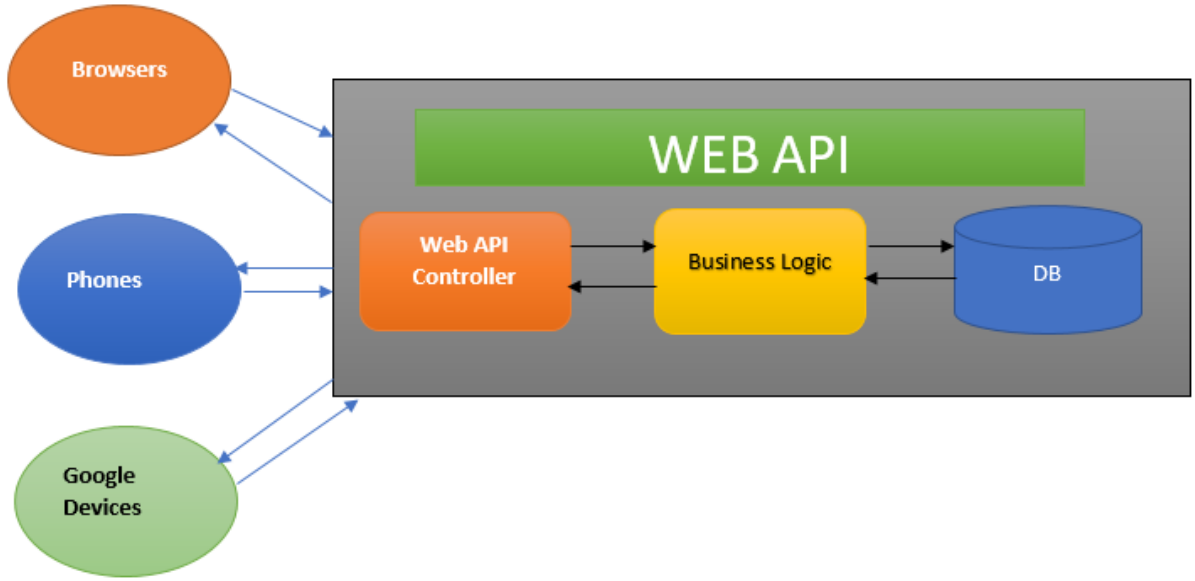
```
[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase
```

Why is Web API required?

The user wants to access the application from different devices like mobile, browser, Google devices, etc. In this case, Web API can be useful.

Different devices request to Web API and Web API will respond in JSON format. Most of the devices are able to understand JSON output.

Web Api Architecture diagram,



This diagram explains the architecture of Web API.

1. A client called api/controller – In the above diagram Browers, Phones, and Google Devices are called Web API Controllers.
2. api/Controller interact with business layer and get Data from DB.
3. The output will be returned in JSON format.