Microsoft

ASP.NET Core

ASP.NET Core

A new open-source and cross-platform framework for building modern cloud-based Web applications using .NET

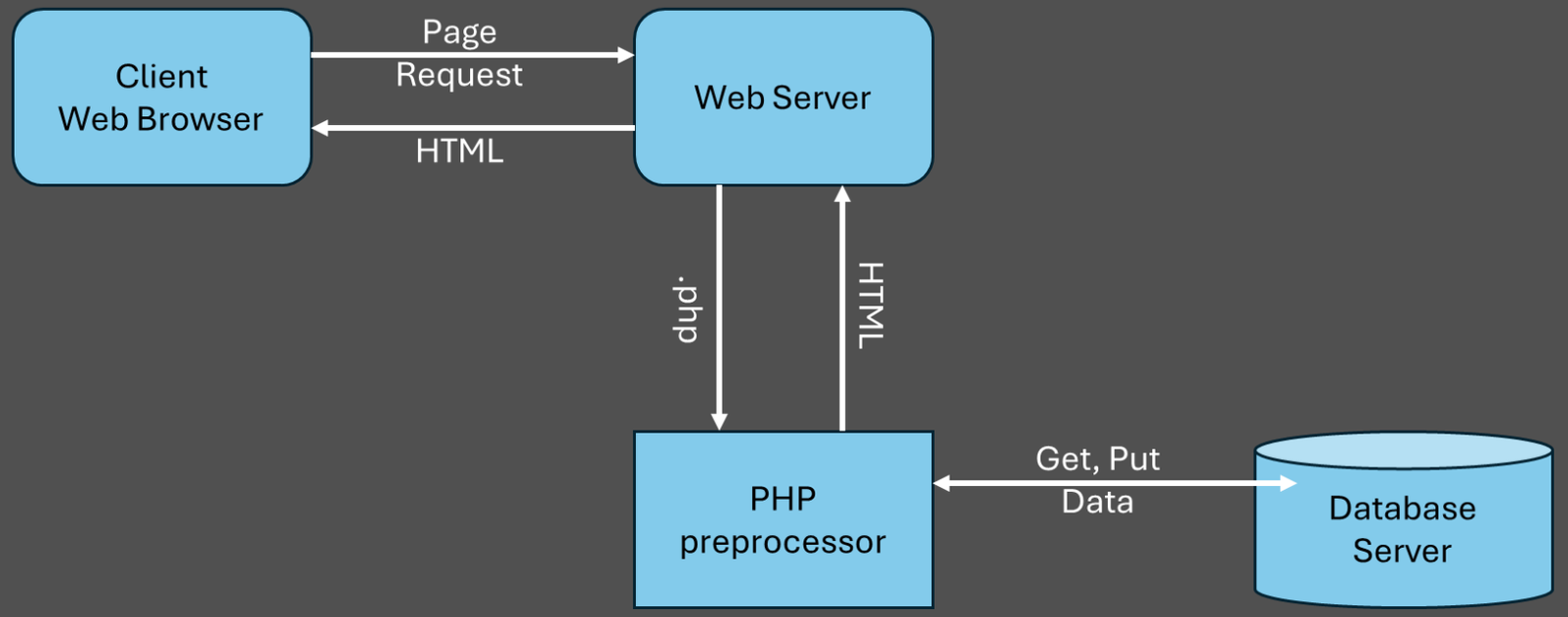
Why ASP.NET Core?

- Unified web stack for web UI and web APIs
- Totally modular
- Microservices & containers
- Cloud ready
- Choose your host
- *Fast*

PHP/JSP/ASP

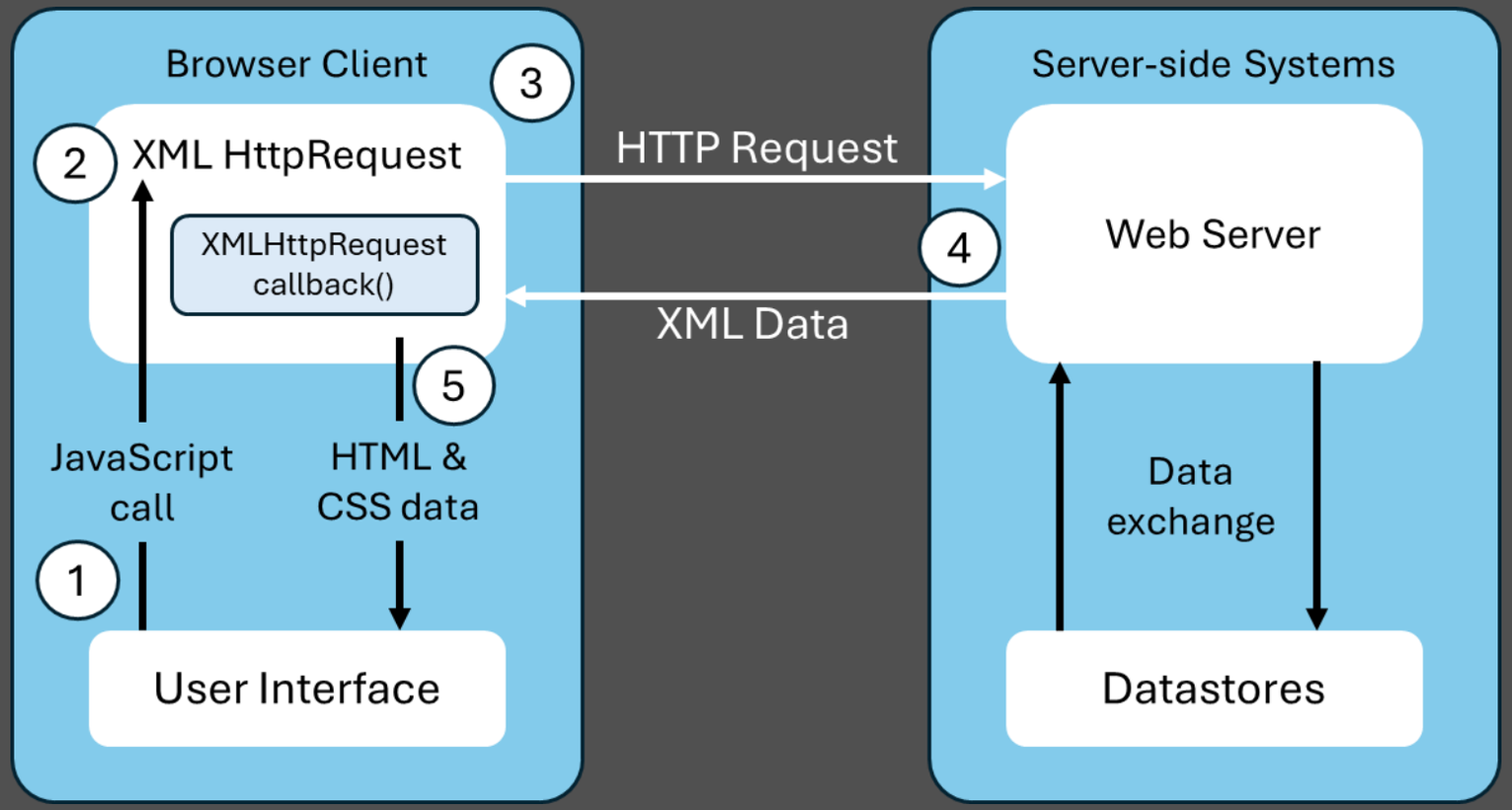
- In the past, all of the front-end code (HTML, JS and CSS) was generated from the back-end.

- User interactions with the webpage often required a full-page refresh.



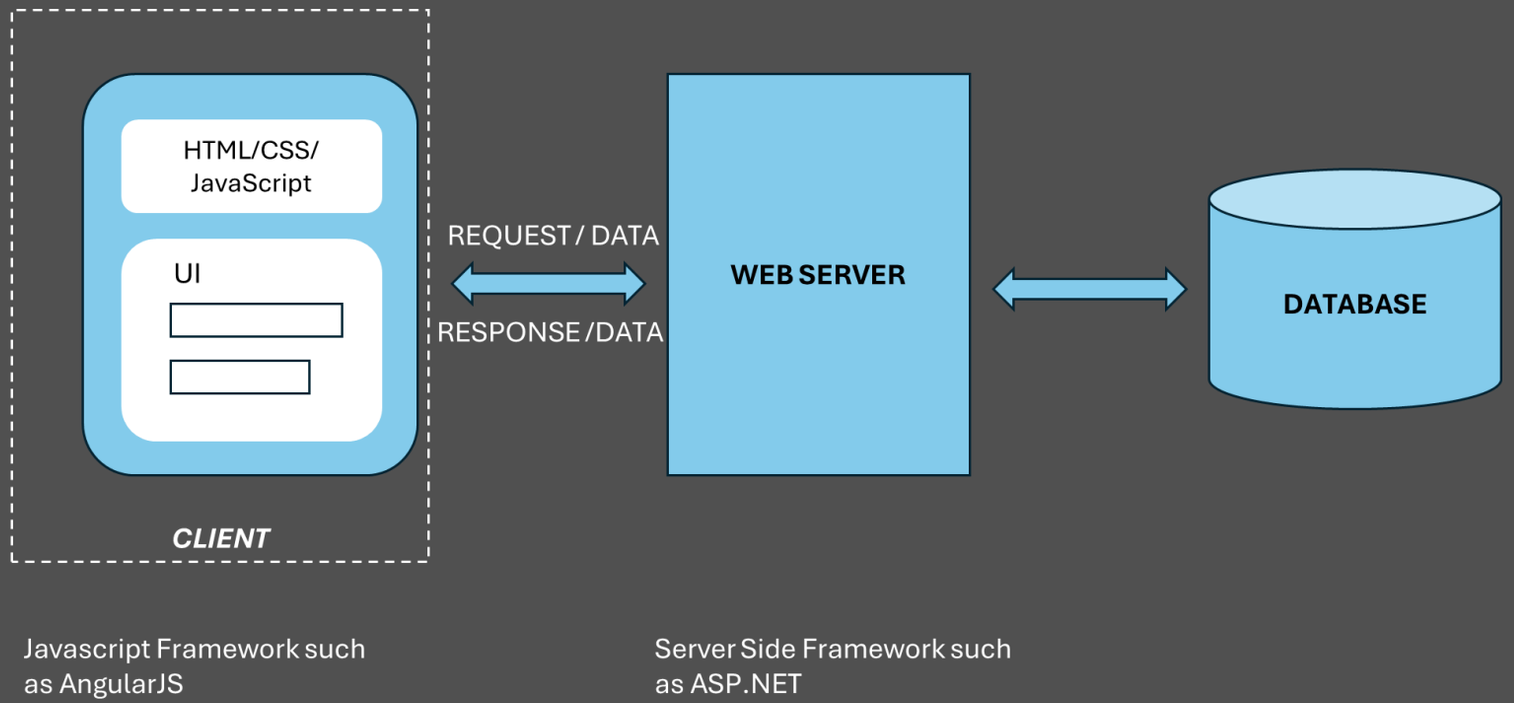
Ajax

- Then came into play AJAX and JQuery.
- The main idea was to load content asynchronously in the background to refresh portions of webpage.

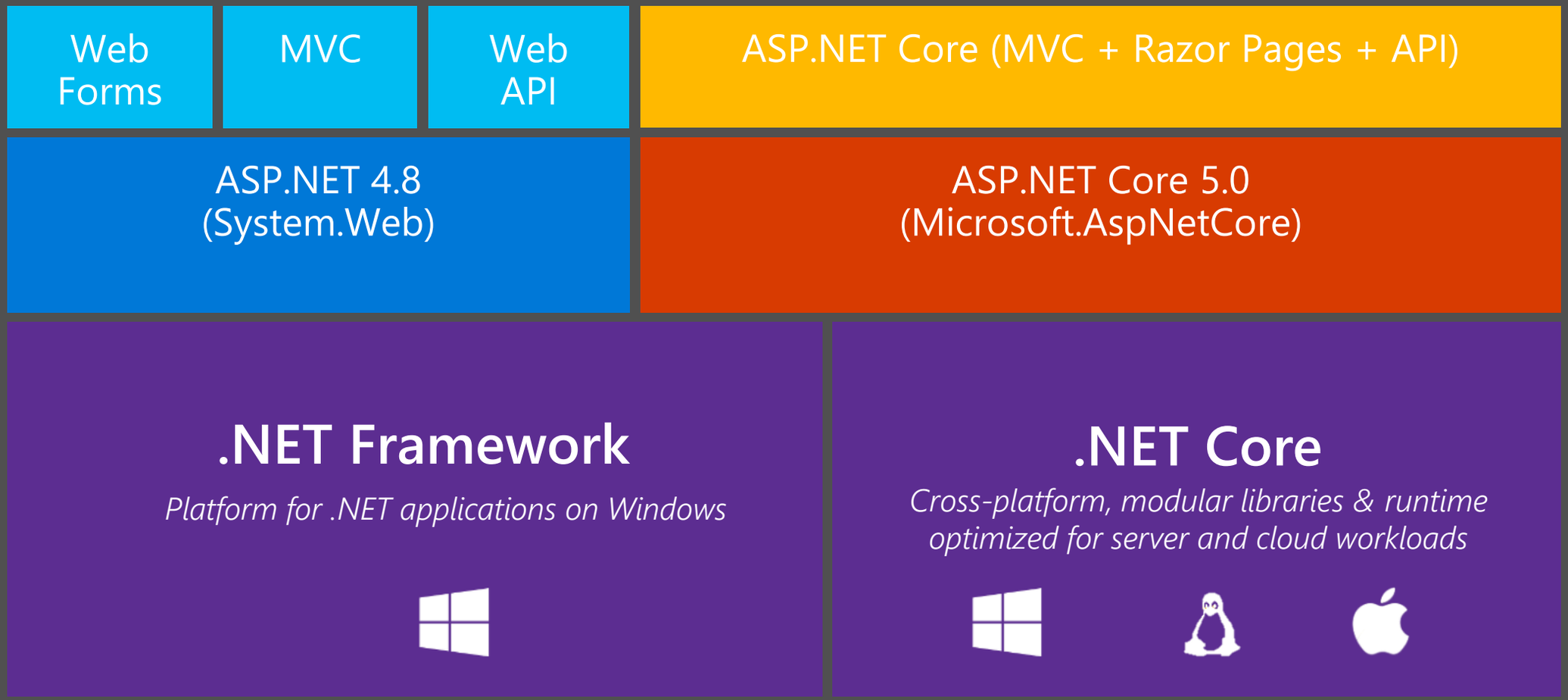


Angular

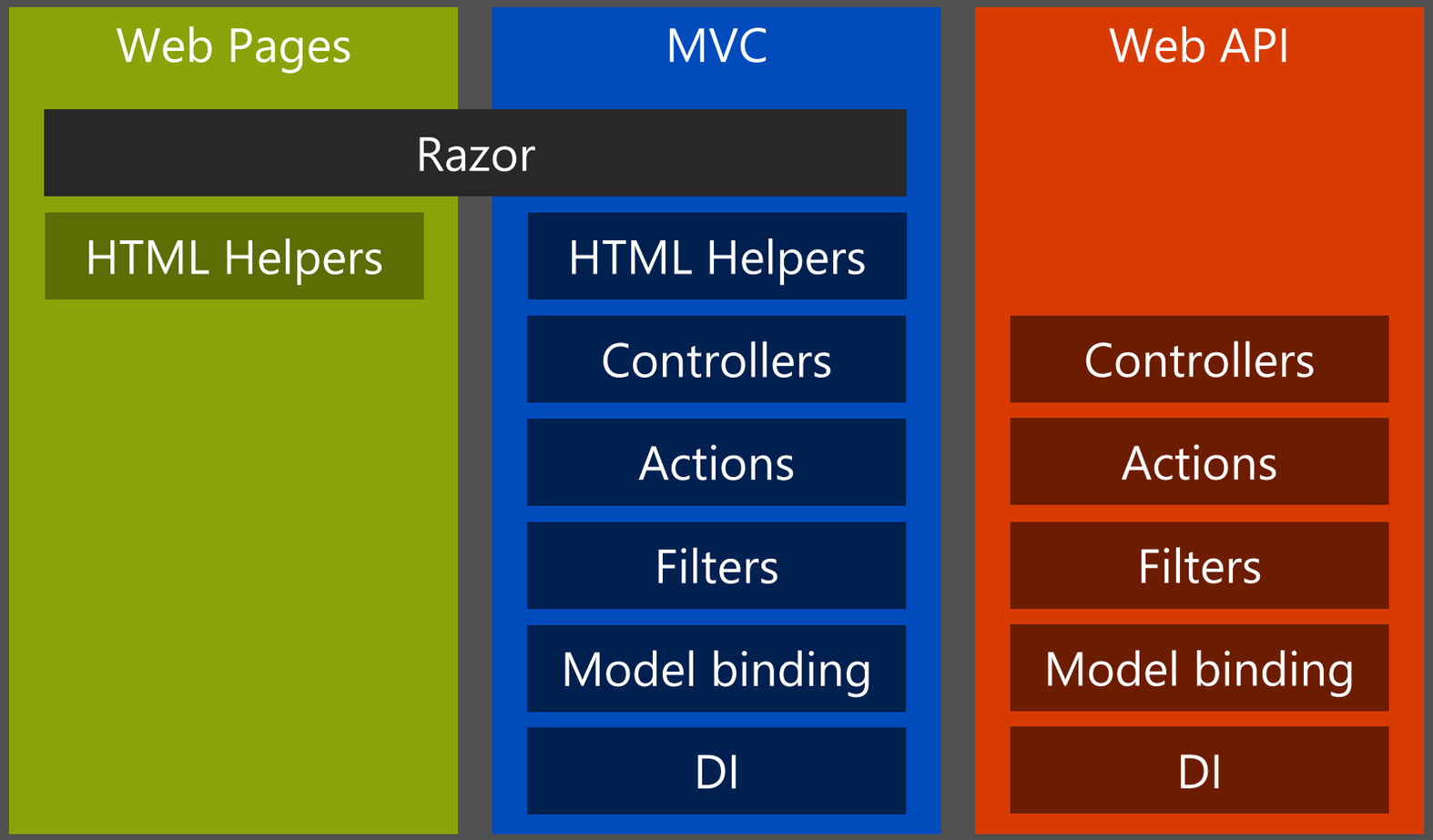
- With Angular, the front-end code is now independent from the back-end.
- The web server becomes a web-service that outputs JSON data, not dynamic HTML or CSS.



ASP.NET Core in a Nutshell



Previous ASP.NET frameworks - similar, but different



ASP.NET = A unified web stack

ASP.NET 4.8
(WebForms, MVC, Web API, Web Pages - present but separate)

- Web Forms
- MVC
- Web Pages
- Web API

ASP.NET Core 1.0
Web API + MVC

ASP.NET Core 2.0
Web API + MVC + Razor Pages

ASP.NET = A unified web stack

ASP.NET 4.6
(WebForms, MVC,
Web API, Web Pages)

ASP.NET Core 1.0
Web API+MVC

ASP.NET Core 2.0
Web API+MVC+Razor Pages

ASP.NET Core vNext
Web API+MVC+Razor Pages
+SignalR

ASP.NET Core features

- Hosting
  - Kestrel, Startup
- Middleware
  - Built-in and custom
- Dependency Injection
- Configuration
- Logging
- Application frameworks
  - MVC, Razor Pages, Identity, SignalR (preview)

ASP.NET Core Benefits and Features

- ASP.NET Core applications can be developed and run across different platforms like
  - Windows, macOS, Linux
- ASP.NET Core applications can be hosted on
  - IIS
  - Apache
  - Docker
  - Self-host in your own process
- One Programming Model for MVC and Web API
- Dependency Injection
- Testability
- Open Source

ASP.NET Core Benefits and Features

- Modular
  - ASP.NET Core Provides Modularity with Middleware Components.
  - Both the request and response pipelines are composed using the middleware components.
  - Rich set of built-in middleware components are provided out of the box.
  - Custom Middleware Components can also be created.

MVC + Web API + Web Pages =

ASP.NET Core MVC

Getting Started with ASP.NET Core

Install .NET Core: https://dot.net/core

Install Visual Studio: https://visualstudio.com

Docs: https://docs.asp.net

Samples and code: https://github.com/aspnet

Your first ASP.NET Core app

- It's just a console app!
- Define your app using **`Startup`**
  - Request handling pipeline:

    `Configure(IApplicationBuilder app)`
  - Services available through DI

    `ConfigureServices(IServiceCollection services)`
- Build your host and run it

Your first ASP.NET Core app

From Visual Studio

File->New project->ASP.NET Core Web App

From .NET Core CLI

```
dotnet new web
dotnet run
```

Your first ASP.NET Core app

Building your host

Build your host using a `WebHostBuilder`

Important host properties and services:

- Server: Which server to use to listen for requests (Kestrel, HTTP.SYS)
- Server URLs: Addresses to listen on
- Environment: Name of current environment (ex Development vs Production)
- Content root: Location of your app content
- Web root: Location of your static assets that should be requestable
- Configuration: Setup available config data
- Logging: Setup logging providers
- Startup: Define your app

Use `WebHost.CreateDefaultBuilder()` to get started fast

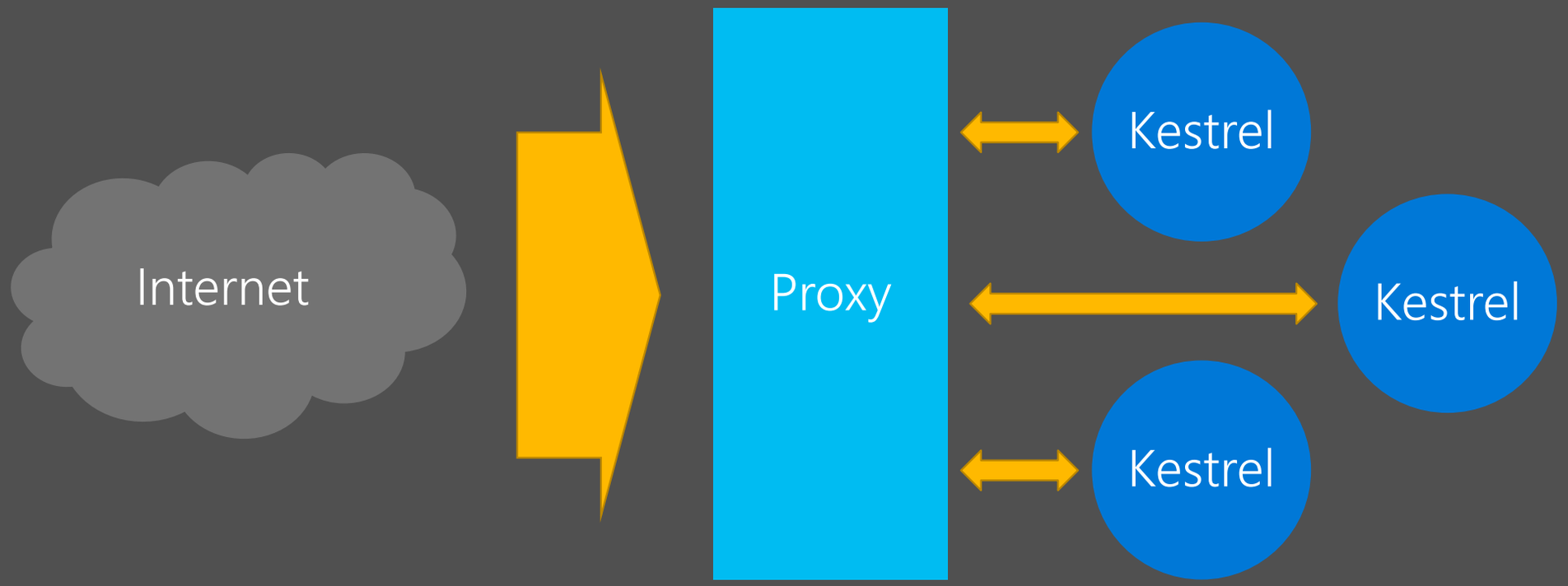
Run the host! (ctrl-c to stop it)

Hosting in production

Built-in web servers – kestrel, HTTP.SYS

Use a reverse proxy for process management

- IIS, Nginx, Apache, HAProxy, etc



ASP.NET Core Project File

- .csproj or .vbproj depending on the programming language used.
- No need to unload the project to edit the project file.
- File or folder references are no longer included in the project file.
- The File System determines what files and folders belong to the project.

ASP.NET Core Project File

- Specifies how the application should be hosted.
  - InProcess or OutOfProcess.

- InProcess hosts the app inside of the IIS worker process(W3Wp.exe).

- OutOfProcess hosting model forward web requests to a backend ASP.NET Core app running the Kestrel server.

- The default is 0ut0fProcess hosting.

ASP.NET Core Project File

- Used to Include a reference to the NuGet package that is installed for the application.
- Metapackage – [MicrosoftAspNetCoreApp] metapackage has no content of its own, it just contains a list of dependencies (other packages).
- When the version is not specified, an implicit version is specified by the SDK.
- Rely on the implicit version rather than explicitly setting the version number on the package reference.

Hosting Types

Some of the Tasks that CreateDefaultBuilder() performs in Main Method.

- Setting up the web server
- Loading the host and application configuration from various configuration sources and Configuring logging

An ASP.NET core application can be hosted

- InProcessor
- OutOfProcess

ASP.NET Core InProcess Hosting

To configure InProcess hosting

```
<AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>
```

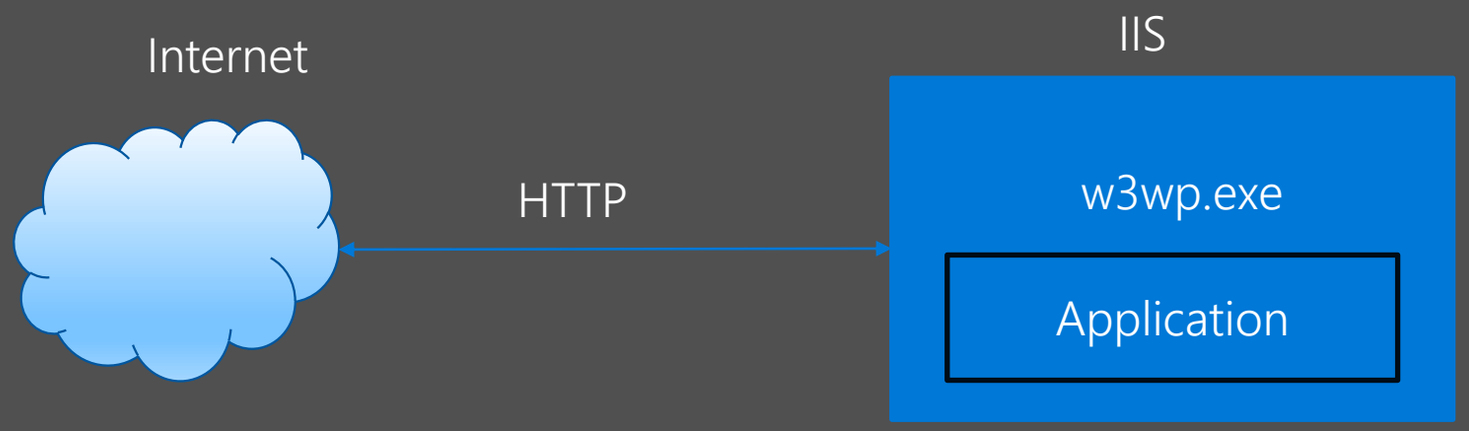
CreateDefaultBuilder() method calls UseIIS() method and host the app inside of the IIS worker process (w3wp.exe or iisexpress.exe).

In Process hosting delivers significantly higher request throughput than OutOfProcess hosting

To get the process name executing the app

```
System.Diagnostics.Process.GetCurrentProcess().ProcessName
```

ASP.NET Core InProcess Hosting



With InProcess hosting

- Application is hosted inside the IIS worker process
- There is only one web server
- From a performance standpoint, InProcess hosting is better than OutOfProcess hosting

ASP.NET Core OutOfProcess Hosting

With Out Of Process hosting

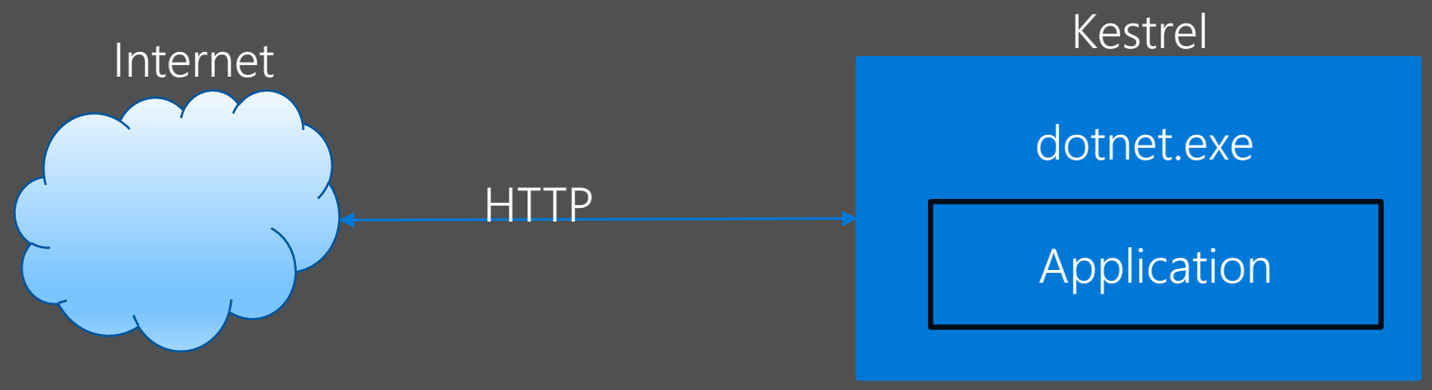
- Web Servers- Internal and External Web Server
- The internal web server is Kestrel
- The external web server can be IIS, Nginx or Apache

What is Kestrel?

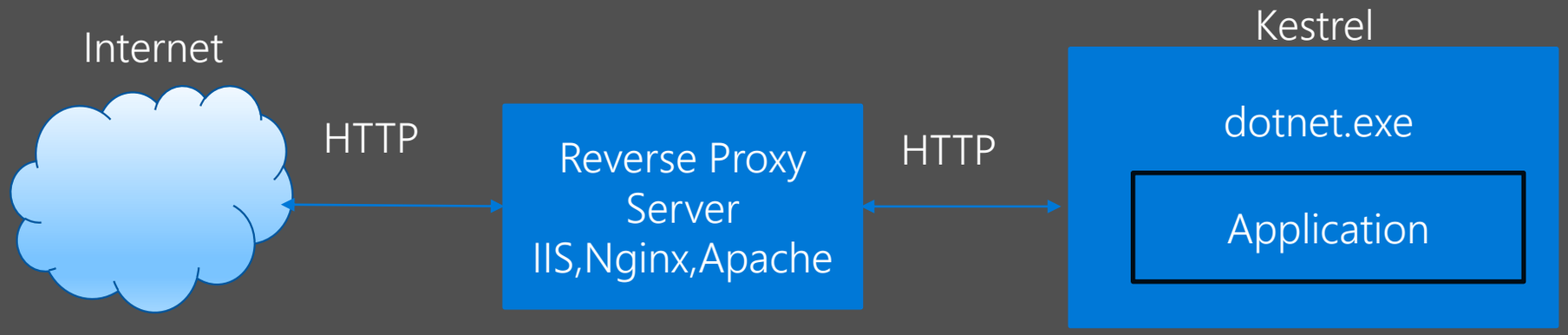
- Cross- Platform Web Server for ASP.NET Core
- Kestrel can be used, by itself as an edge server
- The process used to host the app is dotnet.exe [ApplicationName.exe]

ASP.NET Core Out Of Process Hosting

- Kestrel can be used as the internet facing web server



- Kestrel can be used in combination with a reverse proxy



In Process v/s Out of Process Hosting

In Process	Out of Process
➢ Process name is w3wp.exe or iisexpress.exe	➢ Process name is dotnet.exe
➢ Only one web server	➢ Two web servers
➢ Better for performance	➢ Penalty of proxying requests between internal and external web server

ASP.NET Core Dev Hosting Config

CommandName	AspNetCoreHostingModel	InternalWebServer	ExternalWebServer
Project	Hosting Setting Ignored	Only one web server - Kestrel	
IISExpress	InProcess	Only one web server - IIS Express	
IISExpress	OutOfProcess	Kestrel	IIS Express
IIS	InProcess	Only one web server - IIS	
IIS	OutOfProcess	Kestrel	IIS

Main method in ASP.NET Core

- A Console application usually has a Main() method.
- Why do we have a Main() method in ASP.NET Core Web application.
- ASP.NET Core application initially starts as a Console application.
- This Main() method configures ASP.NET Core and starts it and at that point it becomes an ASP.NET Core web application.

Main method in ASP.NET Core

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }


      public static IWebHostBuilder  CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

Startup Class in ASP.NET Core

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    { }


    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }


        app.Run(async (context) =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    }
}
```

Dependency injection (DI)

Services add in `ConfigureServices` are available through DI

- Inject services in middleware, controllers, page models, filters, views, razor pages, other services

Service lifetimes

- Singleton: There can be only one
- Scoped: One per request
- Transient: A new one every time

Simple built-in IoC container supports only ctor injection

- Replace built-in IoC container with your preferred container

Host provided services

`IHostingEnvironment`: Access host properties

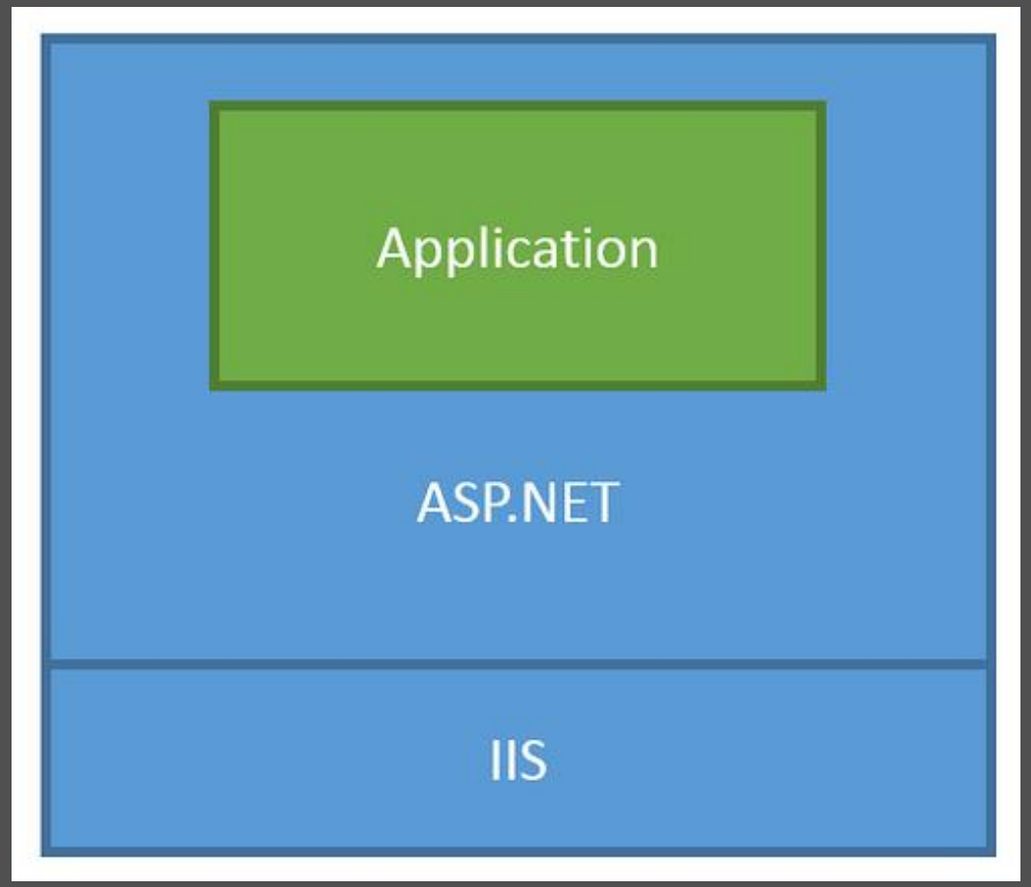
`ILogger<T>`: Logger for the current type

`IConfiguration`: Configuration data

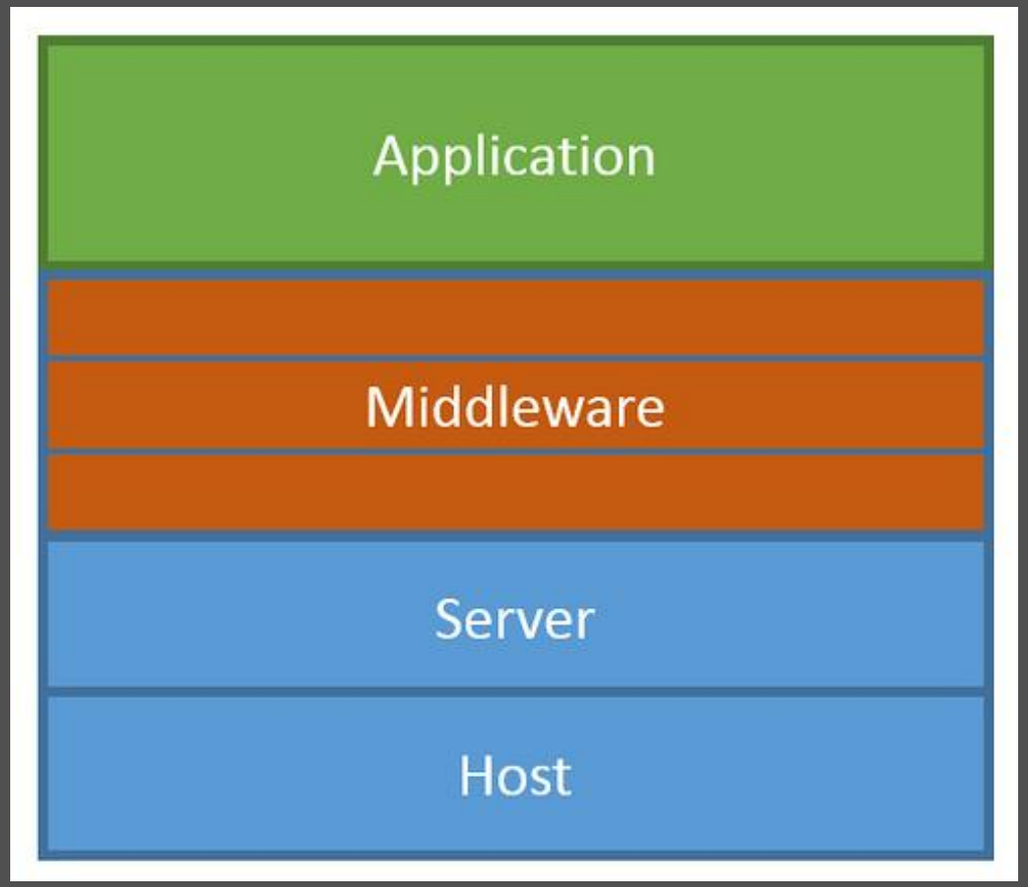
Configuration, Logging, Environments

ASP.NET Core Middleware

Traditional ASP.NET app model



ASP.NET Core middleware

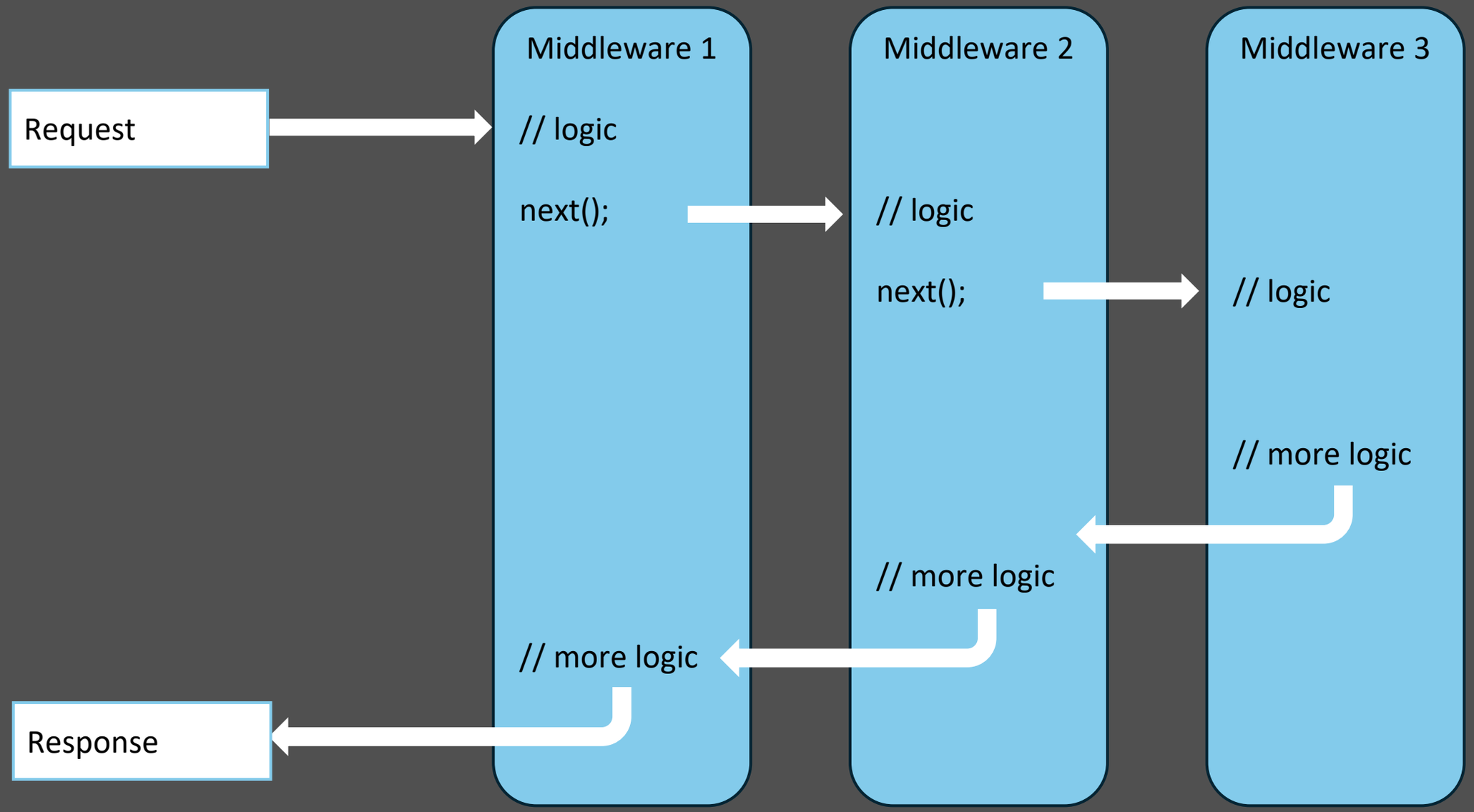


Middleware in ASP.NET Core

Middleware in ASP.NET Core

- Has access to both Request and Response
- May simply pass the Request to next Middleware
- May process and then pass the Request to next Middleware
- May handle the Request and short-circuit the pipeline
- May process the outgoing Response
- Middleware's are executed in the order they are added

ASP.NET Core middleware



Built-in middleware

Routing
Authentication
Static files
Diagnostics
Error handling
Session
CORS
Localization

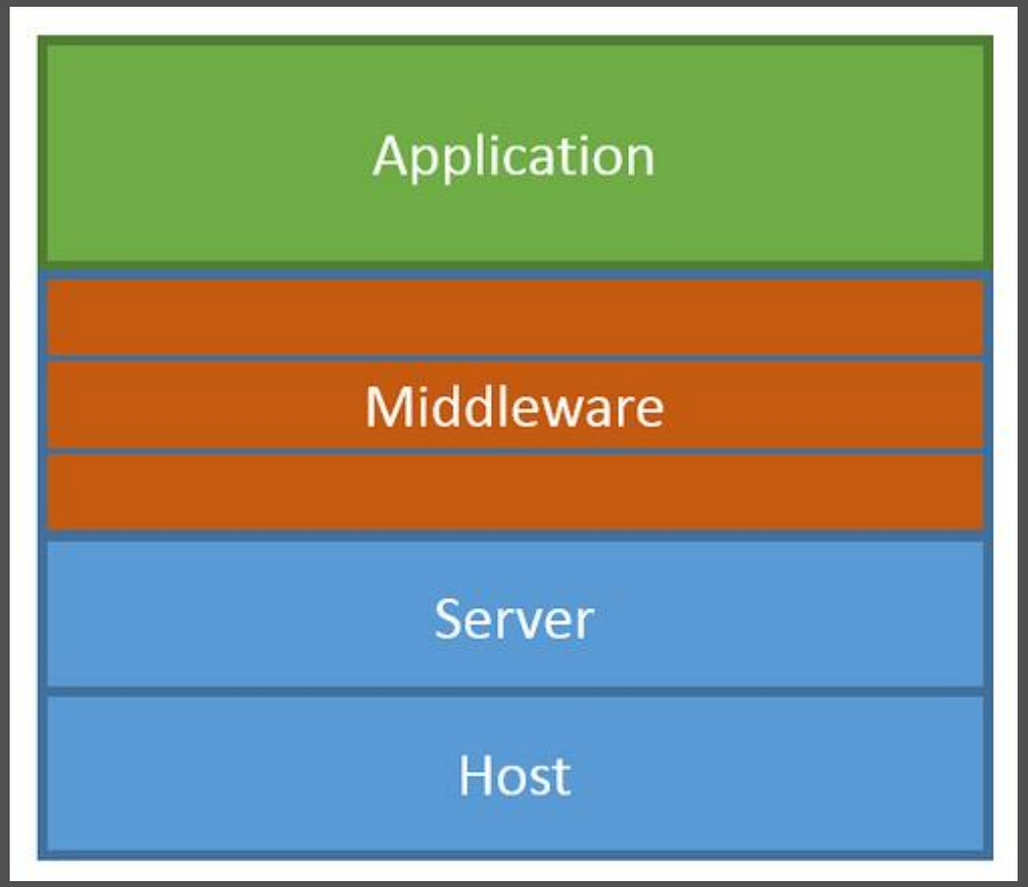
Response compression
Response caching
Forwarded headers
HTTP method overrides
WebSockets
URL rewrite
More to come!

Middleware

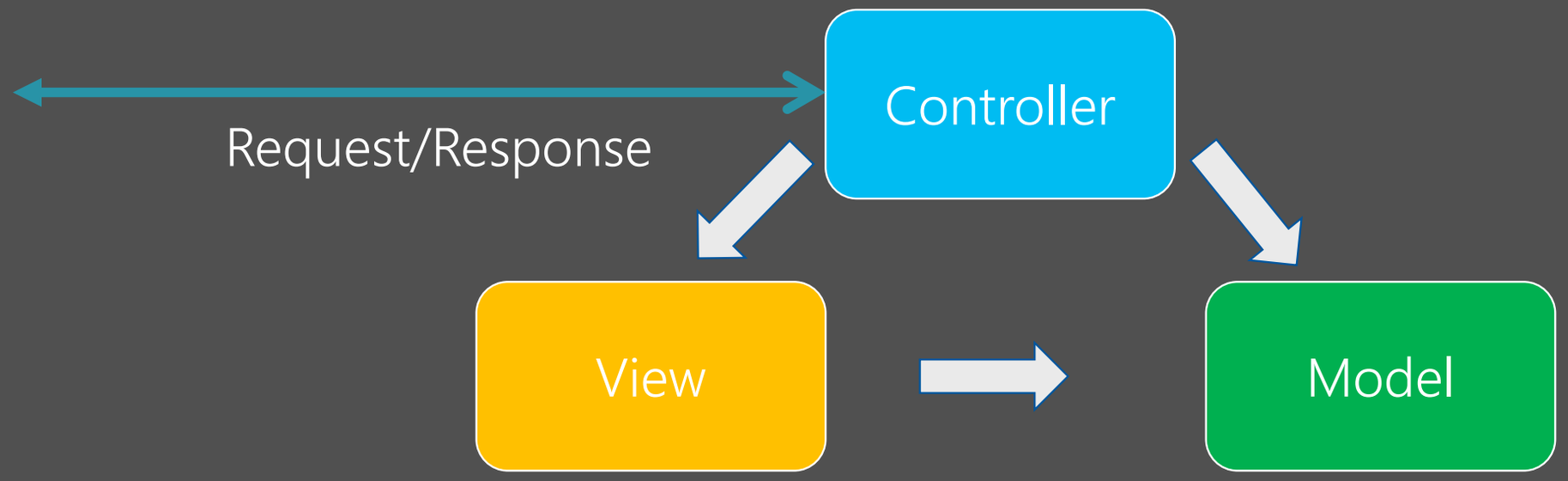
ASP.NET Core MVC

ASP.NET Core MVC

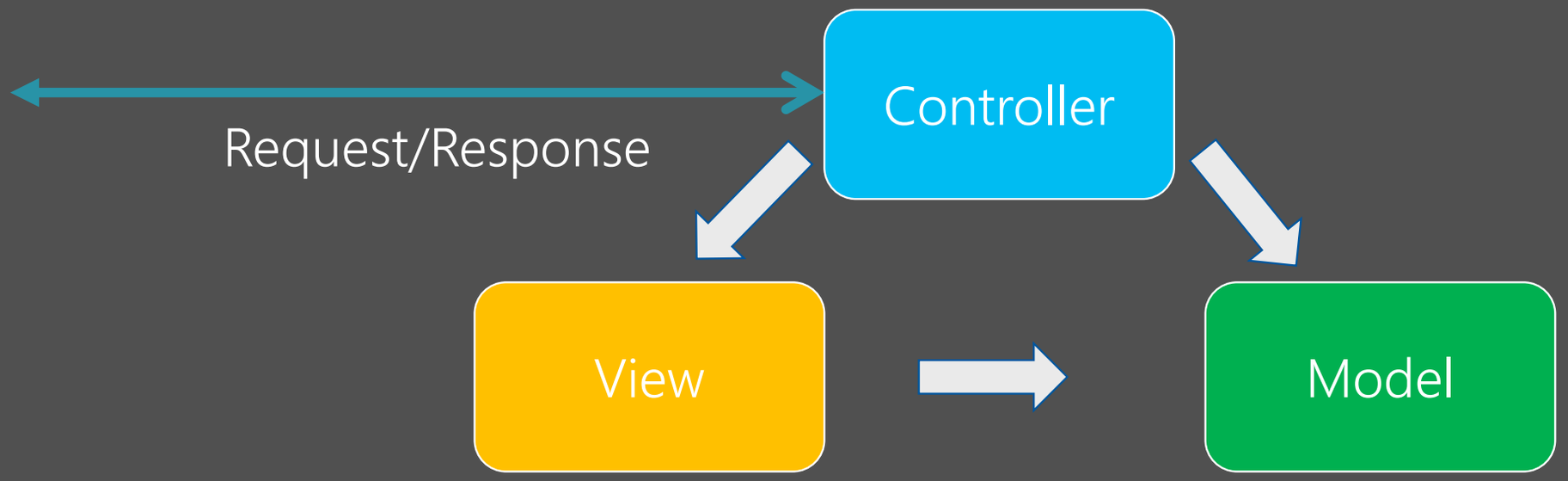
ASP.NET Core middleware

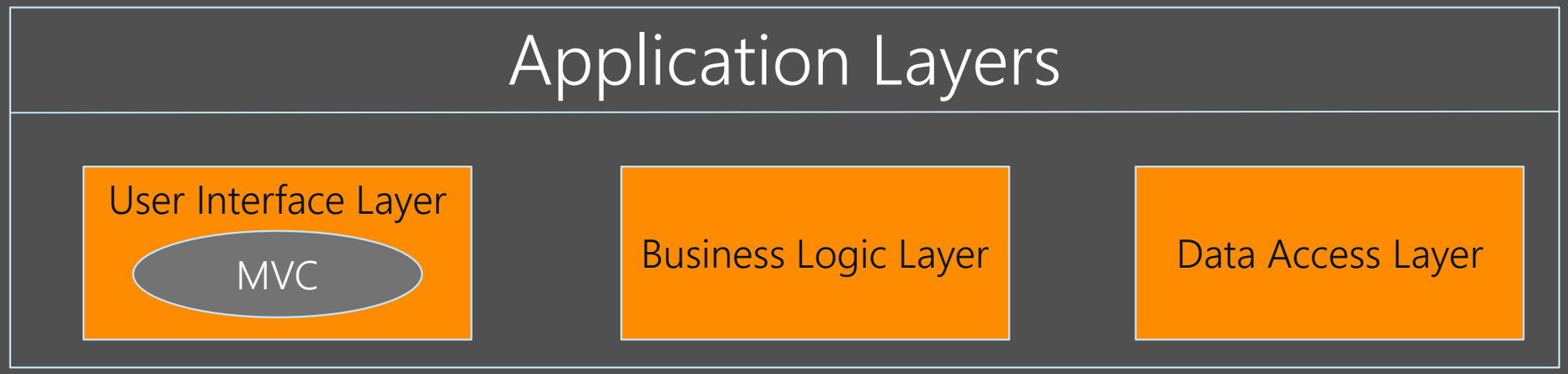


What is MVC?

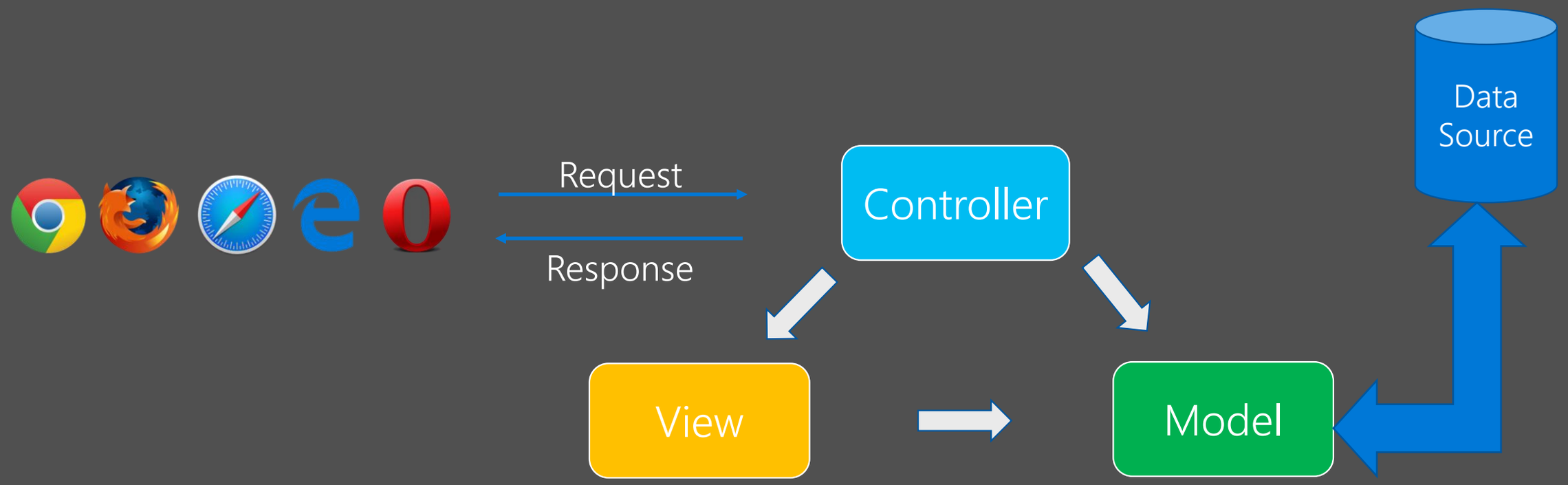


What is MVC?

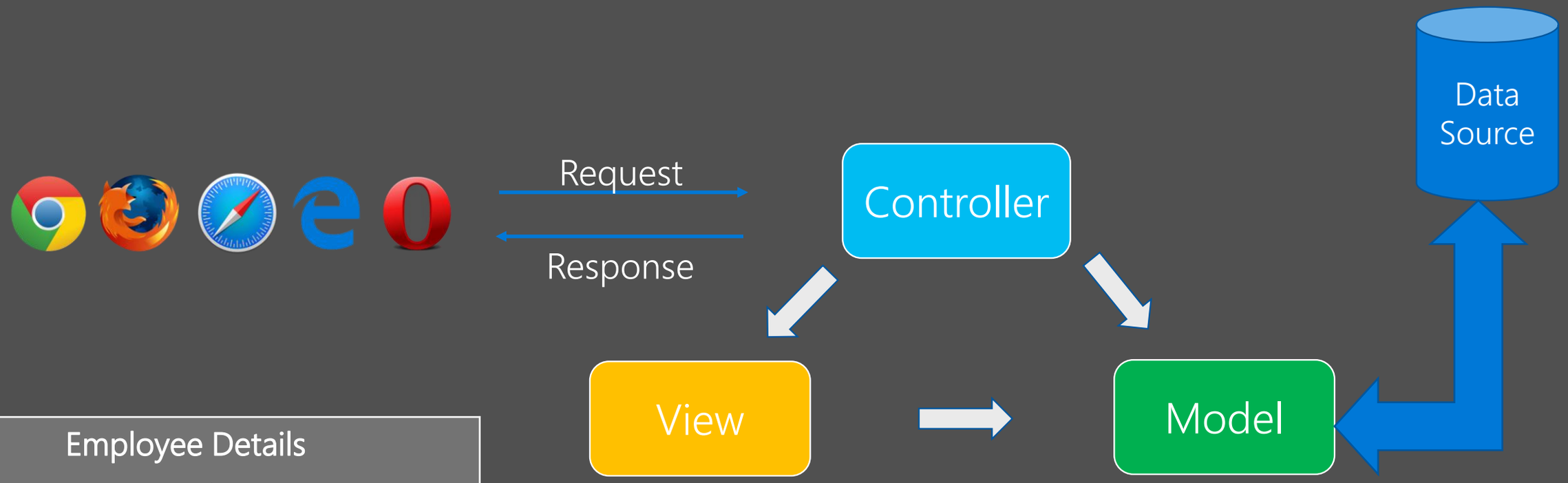

Controller
Request/Response
View
Model


Application Layers
User Interface Layer
MVC
Business Logic Layer
Data Access Layer

How MVC Works



How MVC Works



Employee Details	
ID	1
Name	John
Department	IT

M – V - C

MVC is an architectural design pattern for implementing User Interface Layer of an application

- Model : Set of classes that represent data + the logic to manage that data
- View: Contains the display logic to present the Model data provided to it by the Controller
- Controller: Handles the http request, work with the model, and selects a view to render that model

Passing Data to View in ASP.NET Core MVC

Different ways of passing data to a View from a Controller

- ViewData
- ViewBag
- Strongly Typed View
  - ViewModel
  - Tuple

ViewData

- Dictionary of weakly typed objects
- Use string keys to store and retrieve data
- Dynamically resolved at runtime
- No compile - time type checking and Intellisens

ViewBag

ViewBag v/s ViewData

- ViewBag Is a wrapper around ViewData
- Creates a loosely typed view
- ViewData uses string keys to store and retrieve data
- ViewBag used dynamic properties to store and retrieve data
- Resolved dynamically at runtime
- No compile - time type checking and intellisense
- Preferred approach to pass data from a controller to a view is
- by using a Strongly Typed View

Strongly Typed View

To create a Strongly Typed View

- Specify model type in the view using @model directive

```
@model EmployeeManagement.Models.Employee
```

- To access the model object properties we use @Model

```
<div>
Name : @Model.Name
</div>
<div>
Email : @Model.Email
</div>
```

- Strongly Typed View provides compile-time type checking and intellisense

ASP.NET Core Model Binding

To bind the request data to the controller action method parameters, model binding looks for data in the HTTP request

Form Values

Route Values

Query Strings

ASP.NET Core Model Validation

Validation framework of .NET Core is configured to get the models validated

Built-in Validation Attributes

- RegularExpression
- Required
- Range
- MinLength
- MaxLength
- Compare

Error Handling

Handling errors like, Exceptions, Custom Exceptions, Resource Not Found can be done in multiple places.

- In Try Catch Block
- Built In Middleware
- Custom Middleware

Logging

ASP.NET Core Logging Provider

By default Asp.Net Core provide Logging Providers like Console, Debug for storing and displaying the log's in different placess

Built In Logging Providers

- Console
- Debug
- EventSource
- EventLog
- TraceSource
- AzureAppServicesFile
- AzureAppServicesFBlob
- ApplicationInsights

Third Party Providers

- NLog
- logstash
- splunk
- JSNLog
- Serilog
- KissLog.net
- Sentry
- Loggr

Log Levels

Log Level indicates the severity of the logged message

Trace=0	LogTrace()
Debug=1	LogDebug()
Information=2	LogInformation()
Warning=3	LogWarning()
Error=4	LogError()
Critical=5	LogCritical()
None=6	N/A

Logs can be filtered by

- Log Category
- Logging Provider
- Even both

```
{
  "Logging": {
    "Debug": {
      "LogLevel": {
        "Default": "Warning",
        "EmployeeManagement.Controllers.HomeController": "Warning",
        "EmployeeManagement.Models.SQLEmployeeRepository": "Warning",
        "Microsoft": "Warning"
      }
    },
    "LogLevel": {
      "Default": "Trace",
      "EmployeeManagement.Controllers.HomeController": "Trace",
      "EmployeeManagement.Models.SQLEmployeeRepository": "Trace",
      "Microsoft": "Trace"
    }
  }
}
```

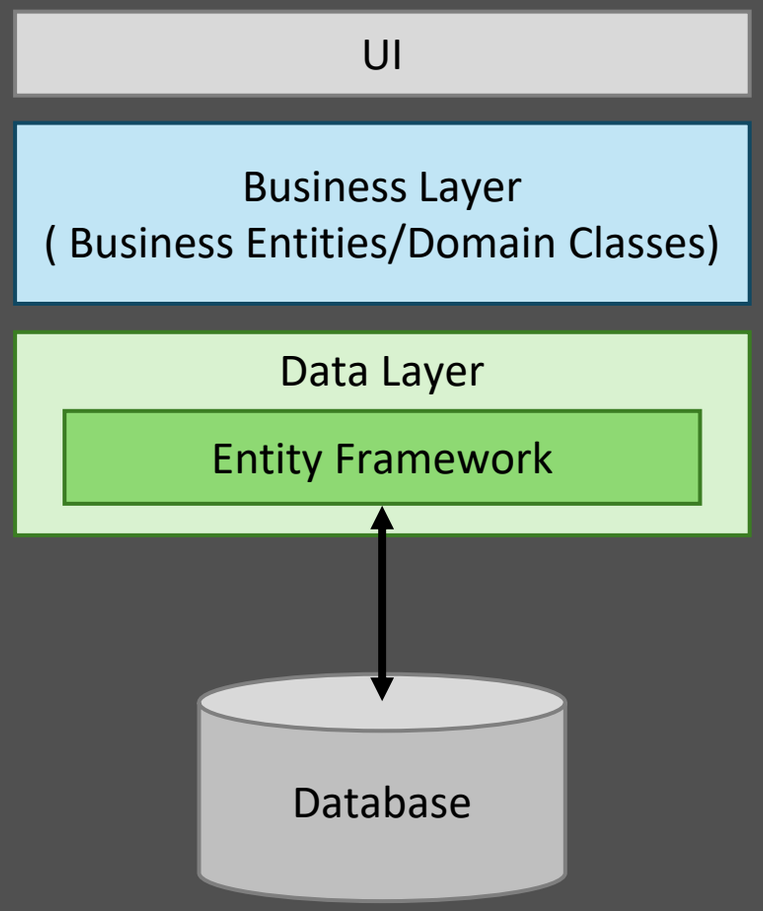
Entity Framework Core

Entity Framework is an object-relational mapper (O/RM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for most of the data-access code that developers usually need to write.

Entity Framework Core

- ORM (Object-Relational Mapper)
- Lightweight, Extensible, and Open Source
- Works Cross Platform
- Microsoft’s Official Data Access Platform

Entity Framework Core



Domain Classes [Entites]

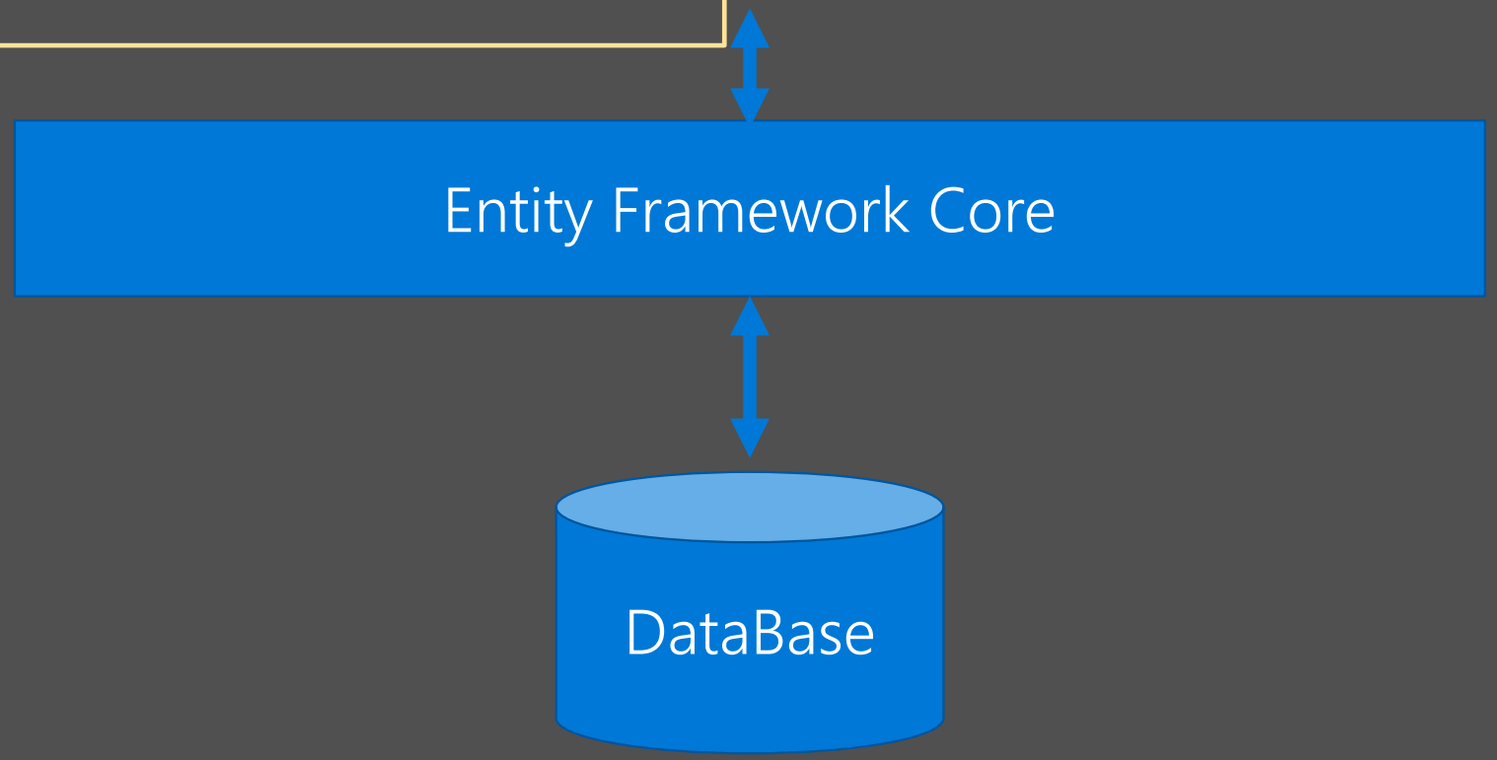
```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public Department Department { get; set; }
}
```

```
public class Department
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Location { get; set; }
}
```

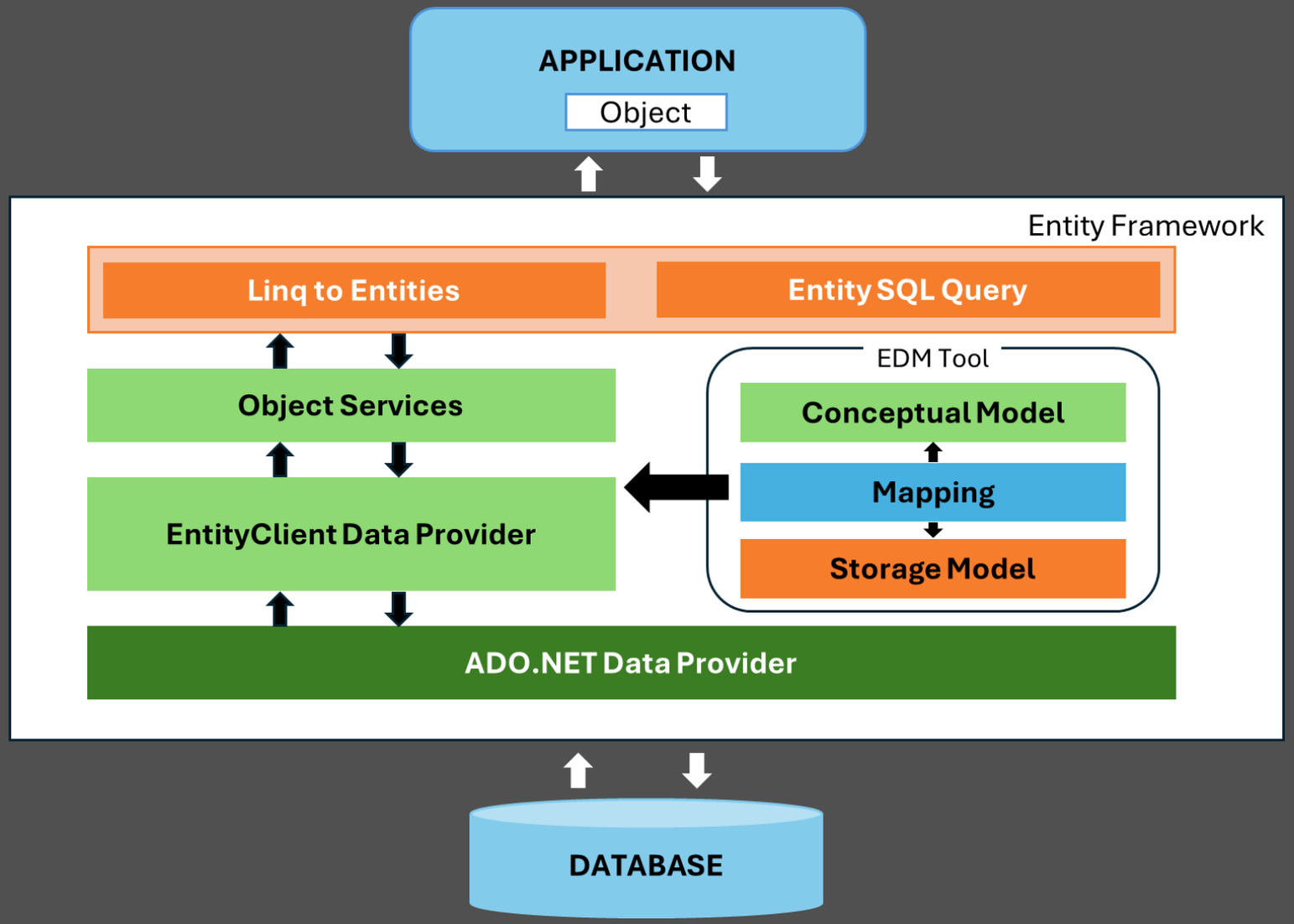
Entity Frame Work Core

```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public Department Department { get; set; }
}
```

```
public class Department
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Location { get; set; }
}
```



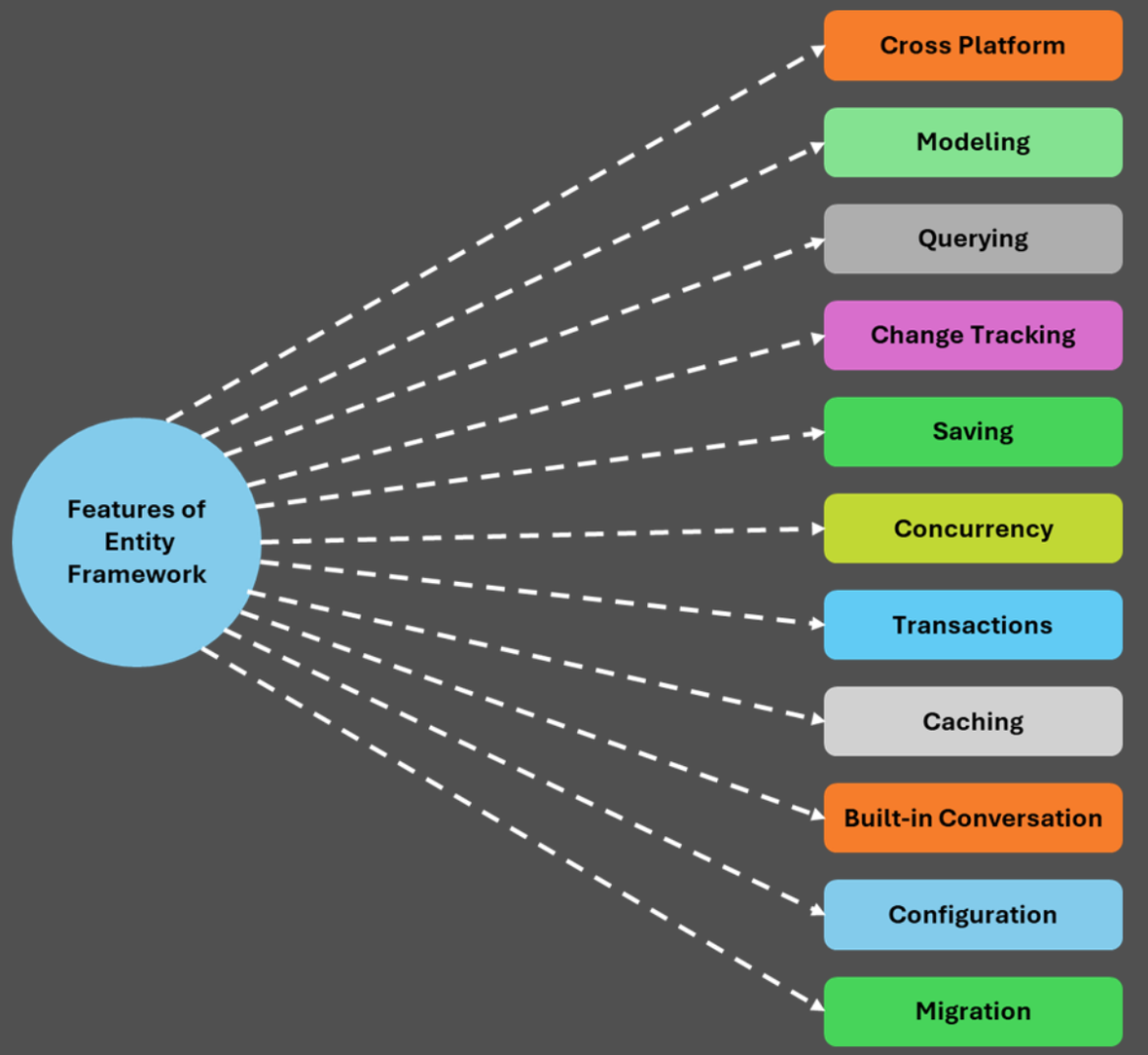
ORM Modeling



Entity Frame Work Approaches

1. Design First
2. Code First
3. Database First

Features



Features of Entity Framework

Benefits

Application Types	ASP.NET Core Applications Web,API, Console, etc…	.NET 4.5+ Applications Console,WinFor, WPF, ASP.NET	Devices + IoT, Mobile, PC, Xbox, Surface Hub	Mobile Application Android,IOS, Windows
EF Core	EF Core	EF Core	EF Core	EF Core
Framework	.NET Core	.NET 4.5+	UWP	Xamarine
OS	Windows, Mac,Linux	Windows	Windows 10	Mobile

LINQ is a 'Language INtegrated Query

LINQ provides the new way to manipulate the data, whether it is to or from the database or with an XML file or with a simple list of dynamic data.

It provides single query interfaces for the different sources of data.

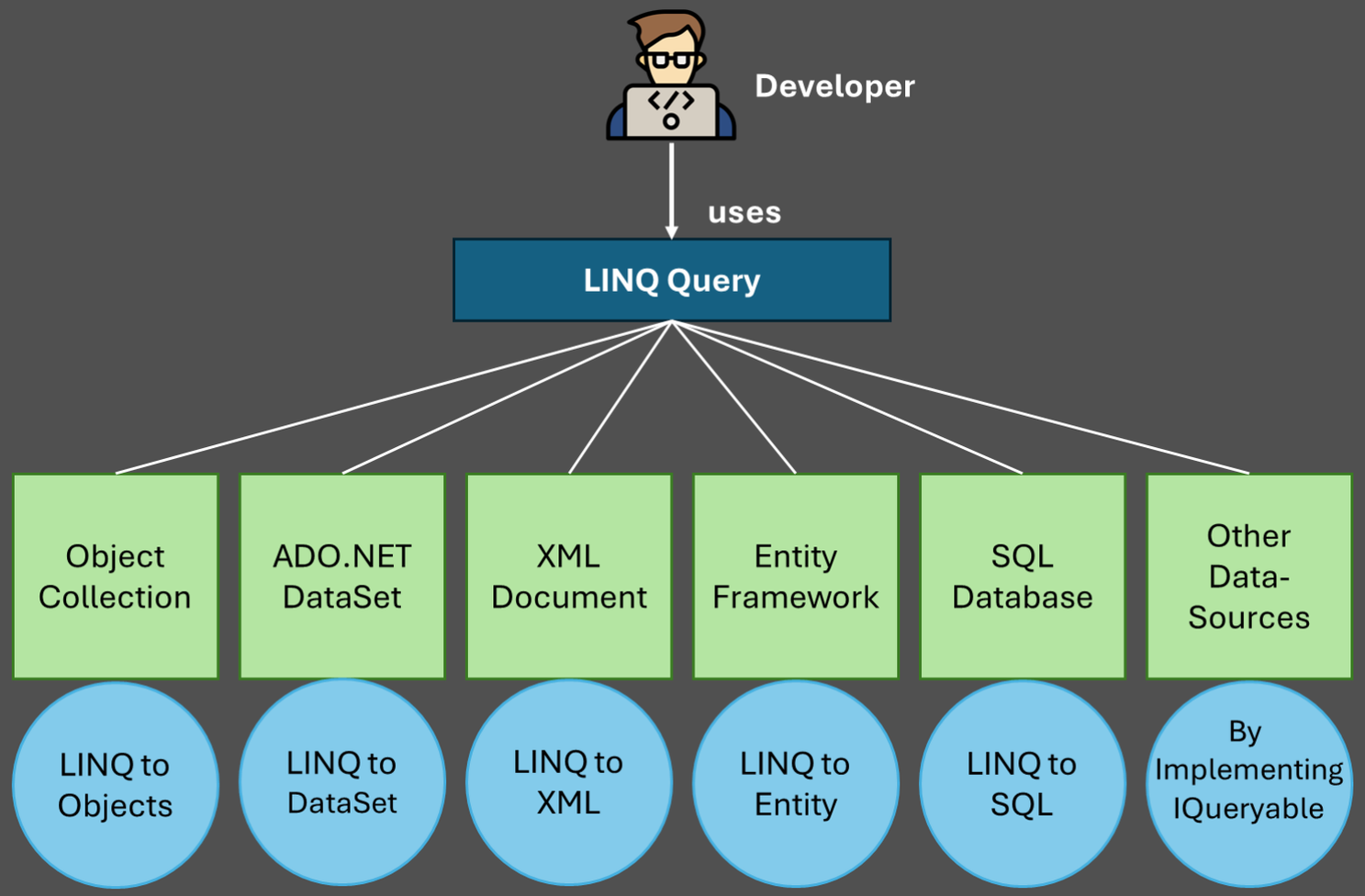
It is integrated with C# to eliminate the mismatch between programming language and database.

LINQ Providers

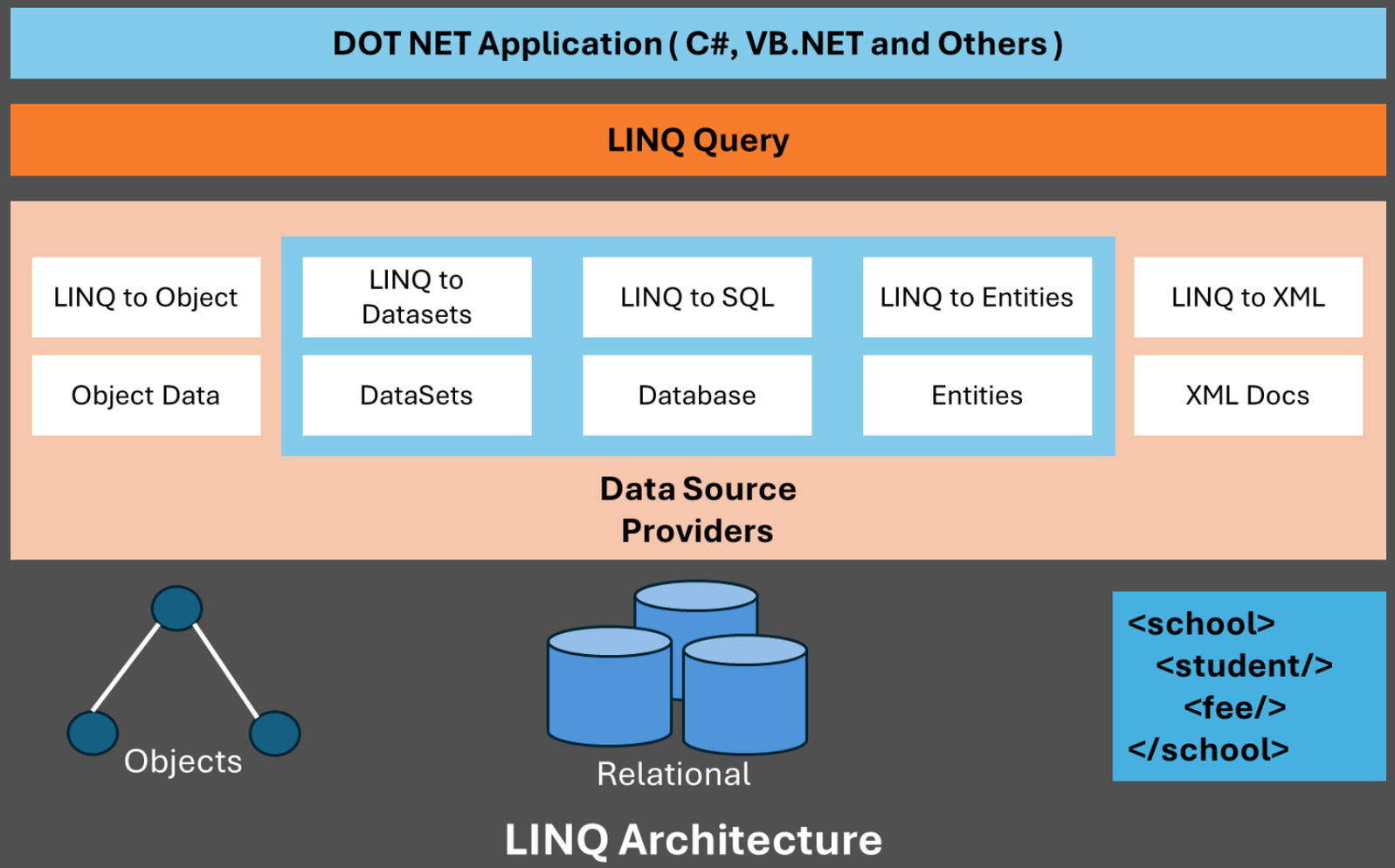
- The .Net Framework has several LINQ providers and facilities:
  - LINQ to Objects
    - Works with IEnumerable<T> collections
  - LINQ to SQL
    - Works with IQueryable<T> data

- Others:

LINQ Providers



Architecture



LINQ Architecture

What is a Query?

A query is nothing but a set of instructions. Queries are applied to the data source to perform the operations (i.e., CRUD operations) and show the shape of the output from that Query

What is a Query?

Each Query is a combination of three things; they are:

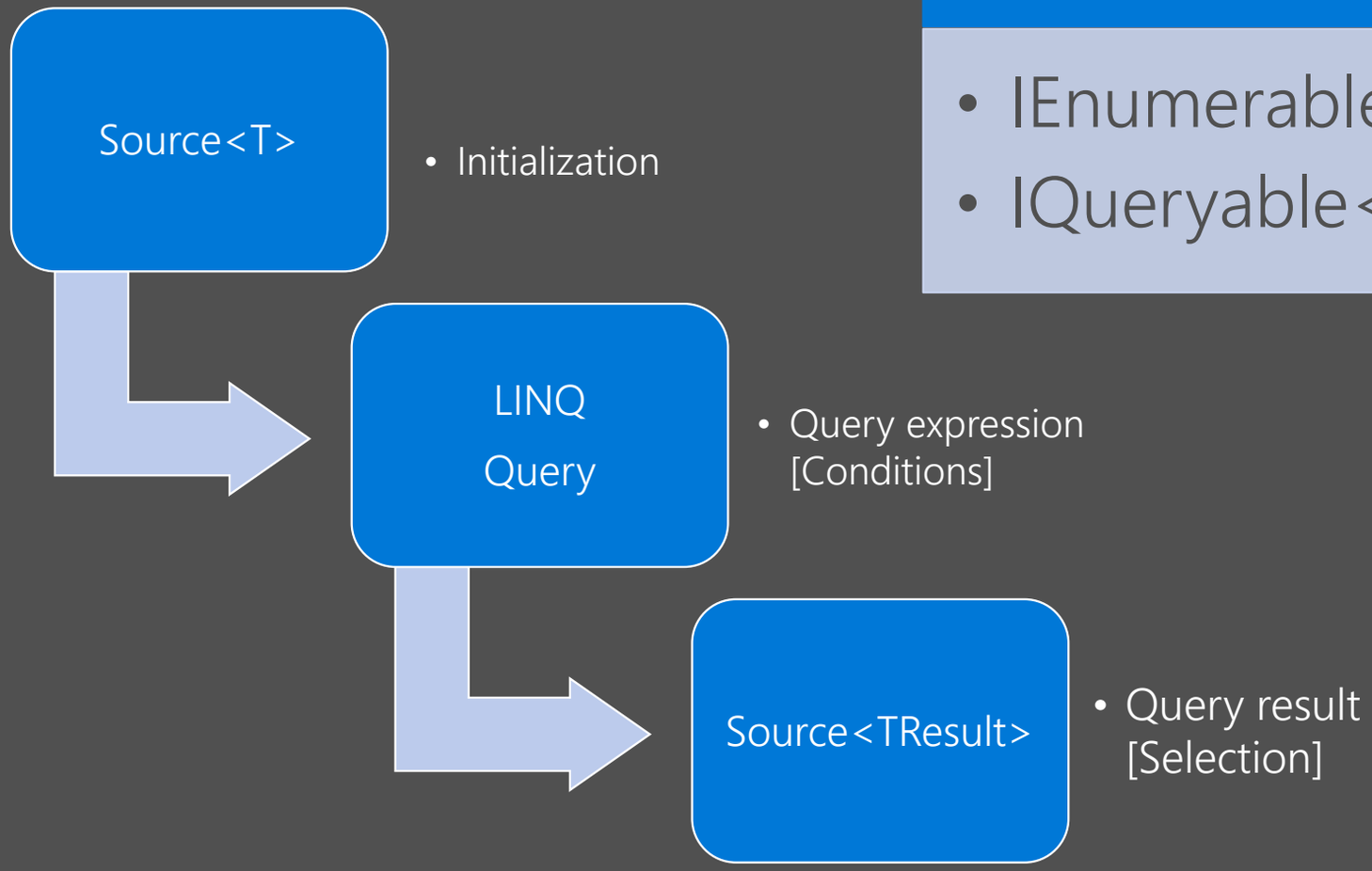
1. Initialization(to work with a particular data source)
2. Condition(where, filter, sorting condition)
3. Selection (single selection, group selection or joining)

Writing Query

We can use LINQ queries in two ways:

1. Query Syntax
2. Method Syntax
3. Mixed Syntax

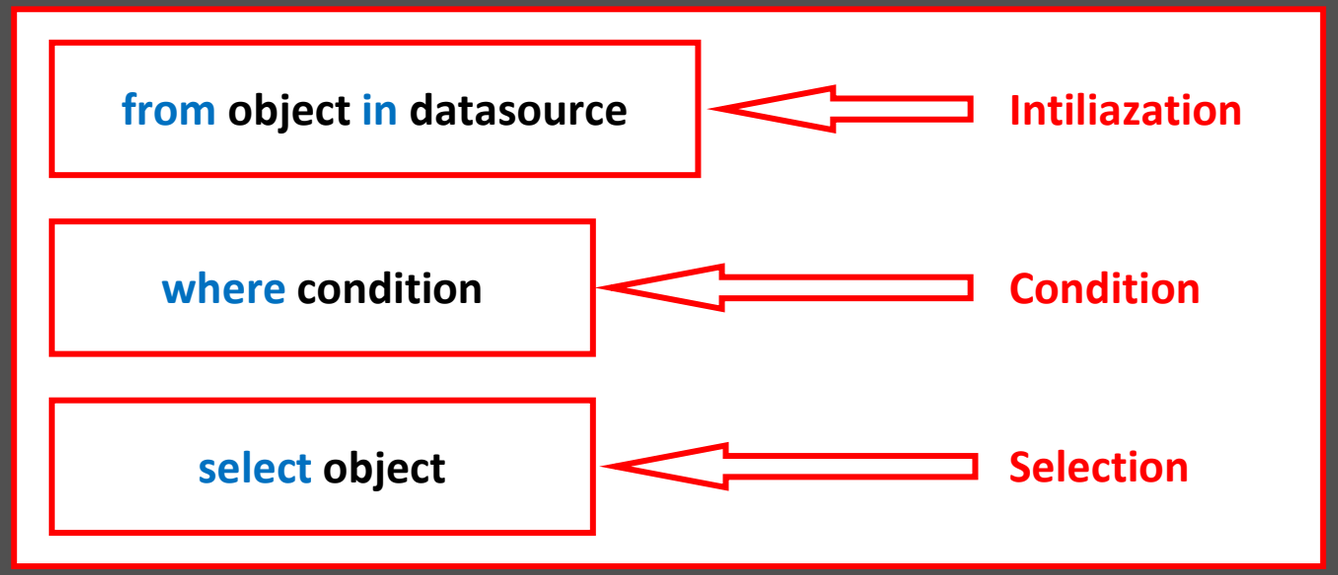
Query Operations



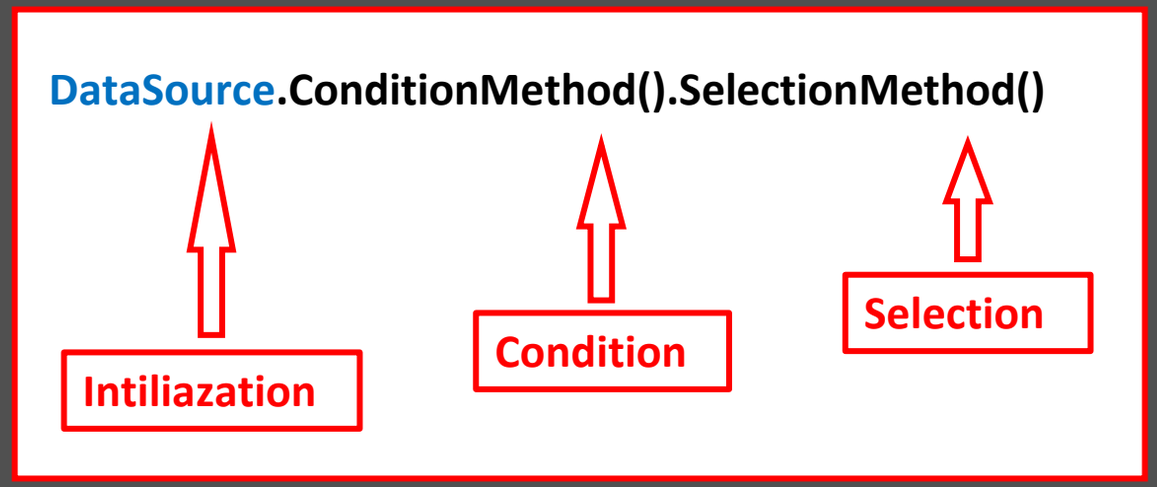
Source

- IEnumerable<T>
- IQueryable<T>

LINQ Query Syntax



LINQ Method Syntax



LINQ Mixed Syntax

```
( from object in DataSource
 where condition
select object).Method()
```

Query Syntax

Method Syntax

LINQ Queries – Extension Syntax

- Here is a typical query using extension method syntax:

```
string[] cars = { "Boxter", "Mini Cooper", "Mustang", "Camaro", "Miata", "Z4" };

IEnumerable<string> query = cars
    .Where(n => n.StartsWith("M"))
    .OrderBy(n => n)
    .Select(n => n);
```

LINQ Queries – Query Syntax

- Here's the same query using Query Syntax:

```
var query =
    from n in cars
    where n.StartsWith("M")
    orderby n
    select n;
```

- Note the difference in capitalization and use of . notation

Query Operators

- Restriction: Where
- Projection: Select, SelectMany
- Partitioning: Take, Skip, TakeWhile, SkipWhile
- Ordering: OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
- Grouping: Groupby
- Set: Distinct, Union, Intersect, Except
- Conversion: ToArray, ToList, ToDictionary, OfType
- Element: First, FirstOrDefault, ElementAt
- Generation: Range, Repeat,
- Quantifiers: Any, All
- Aggregate: Count, Sum, Min, Max, Average, Aggregate
- Miscel: Concat, EqualAll
- Join: Cross Join, Group Join, Cross Join with Group Join, Left Outer Join
- Execution: Deferred Execution, immediate Execution, Query Reuse

LINQ to XML

- LINQ to XML is not based on a LINQ Provider of extension methods.

- LINQ to XML uses the XDocument class that builds and interrogates an XML parse tree.
  - Many of its methods return IEnumerable<T> objects
  - That supports LINQ to Objects queries

- The most important classes are:
  - XDocument
  - XElement
  - XAttribute

ASP.NET Core Security

ASP.NET Core provides many tools and libraries to secure your apps including built-in identity providers, but you can use third-party identity services such as Facebook, Twitter, and LinkedIn.

What we will Learn

Identity Management

Authentication, Role-based and Claim-based authorization

Token Based Auth

Use OAuth to protect Api's

Common Attacks

Understand security risks and how to stop them

Authentication vs. Authorization

- Authentication is a process in which a user provides credentials that are then compared to those stored in an operating system, database, app or resource

- If they match, users authenticate successfully, and can then perform actions that they're authorized for, during an authorization process

- The Authorization refers to the process that determines what a user is allowed to do

Identity Role and Claim

Identity

- Who you are

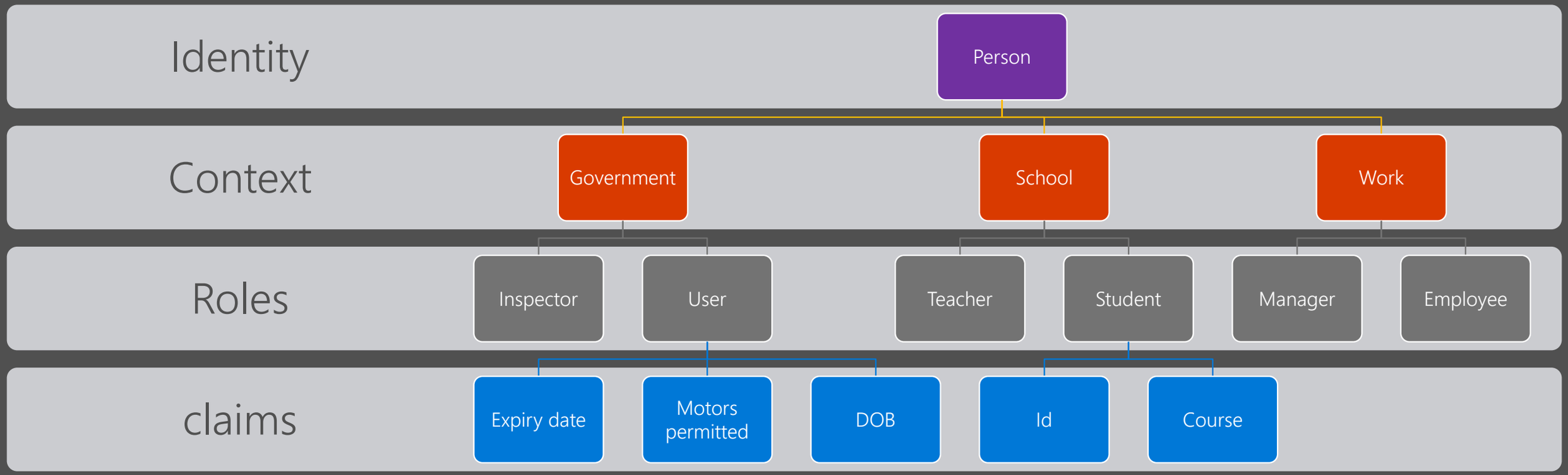
Role

- Who you are in a given context

Claim

- A piece of information about you

Example

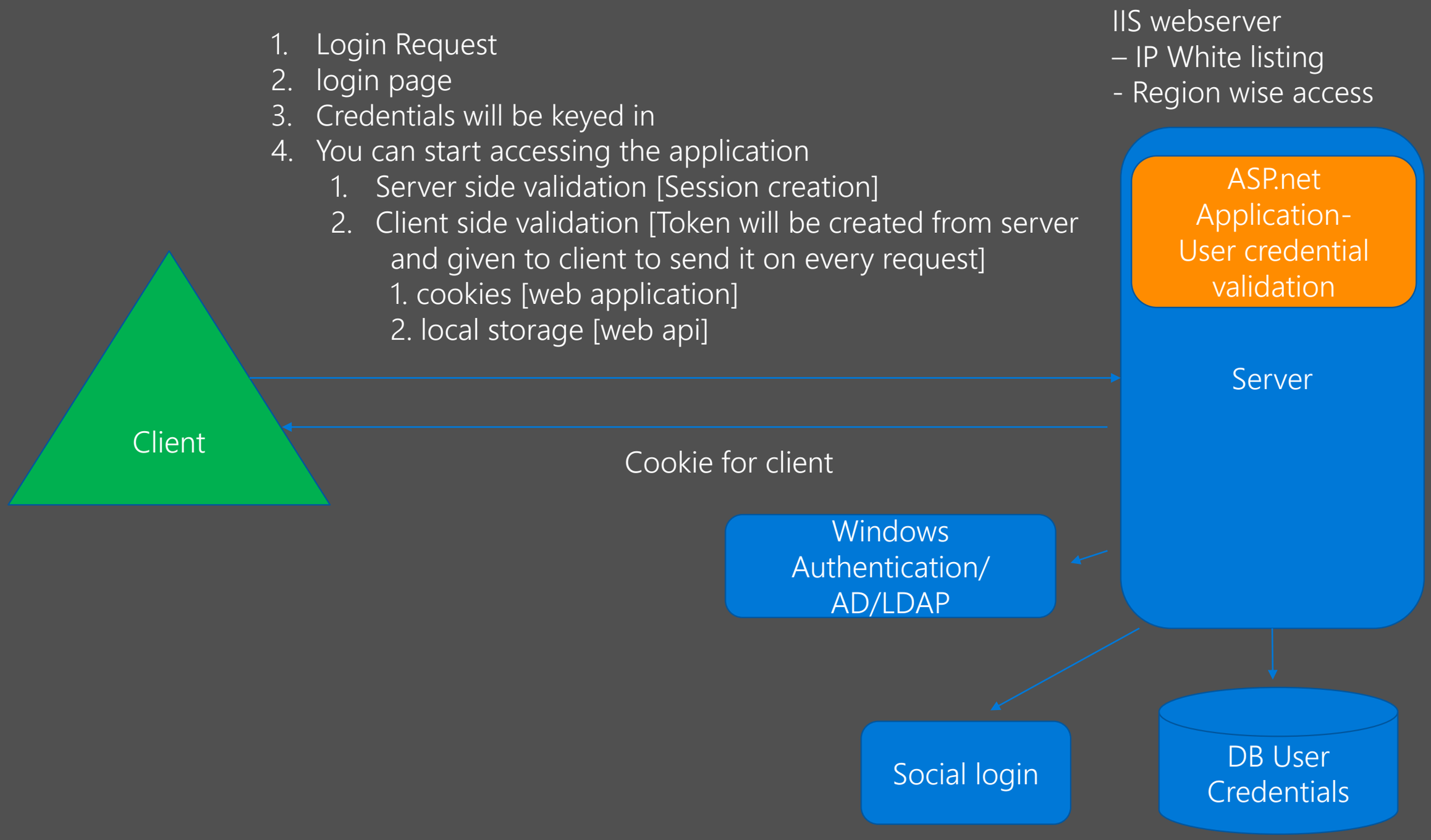

Identity
Person
Context
Government
School
Work
Roles
Inspector
User
Teacher
Student
Manager
Employee
claims
Expiry date
Motors permitted
DOB
Id
Course

Types of Authentication

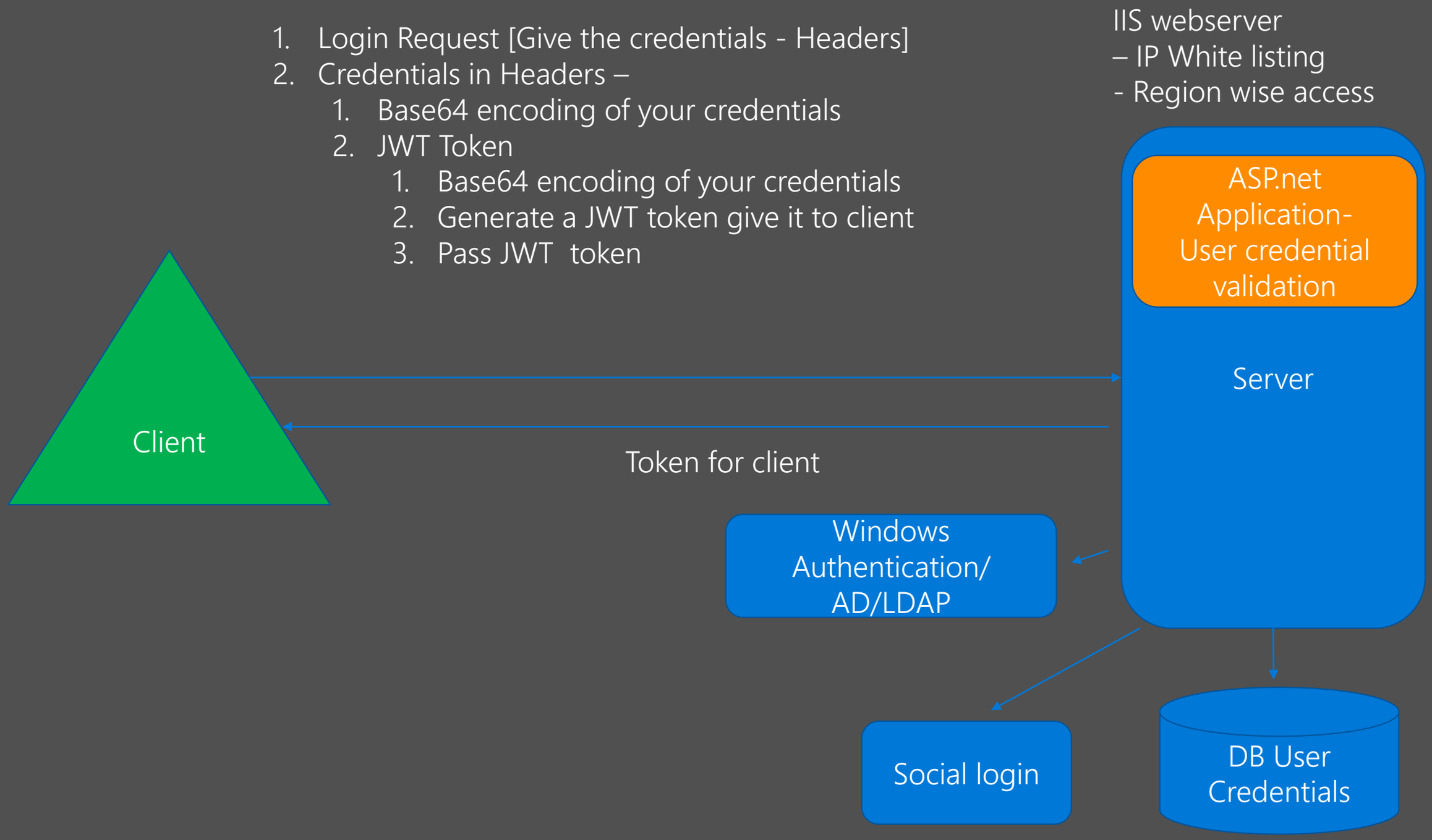
- Cookie Authentication
  - Users Roles and Claims are stored in and managed by the application
- Token Based Authentication
  - Users and Claims are managed by OAuth provider example: LinkedIn, Google, Facebook etc….

Client Server Authentication process

1. Login Request
2. login page
3. Credentials will be keyed in
4. You can start accessing the application
    1. Server side validation [Session creation]
    2. Client side validation [Token will be created from server and given to client to send it on every request]
        1. cookies [web application]
        2. local storage [web api]



1. Login Request [Give the credentials - Headers]
2. Credentials in Headers –
    1. Base64 encoding of your credentials
    2. JWT Token
        1. Base64 encoding of your credentials
        2. Generate a JWT token give it to client
        3. Pass JWT token



Types of Authentication

In ASP.NET Core

Mobile Apps and Javascript

Token Auth

Users and claims are managed by OAuth provider e.g. Google

Server – To – Server Auth

Cookie Auth

Users, Roles and Claims are stored in and managed by the application

Microsoft SQL Server

AWS Cognito

3rd party Auth Provider e.g. Auth()

IdentityUser

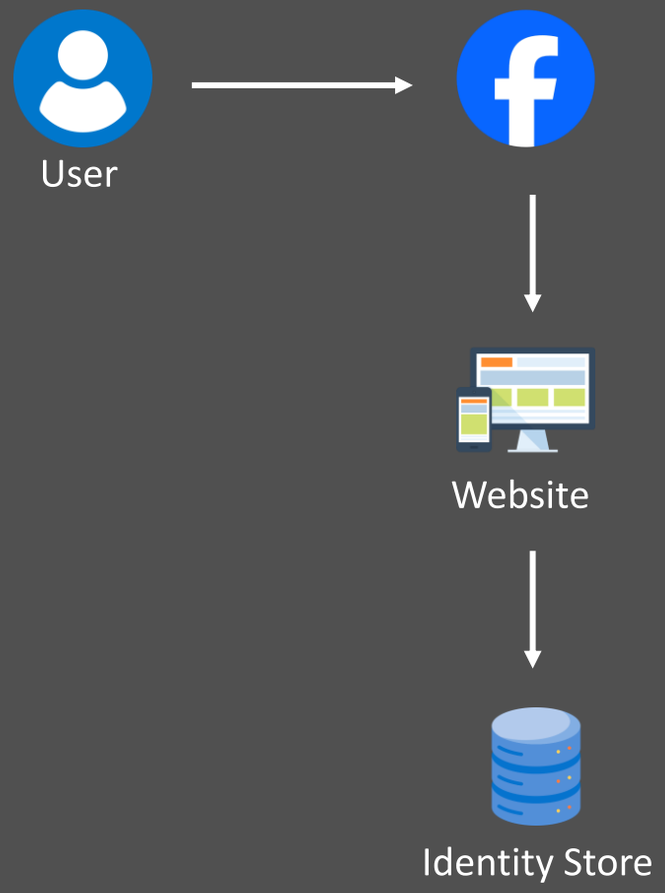
IdentityRole

IdentityUserRole

IdentityUserClaim

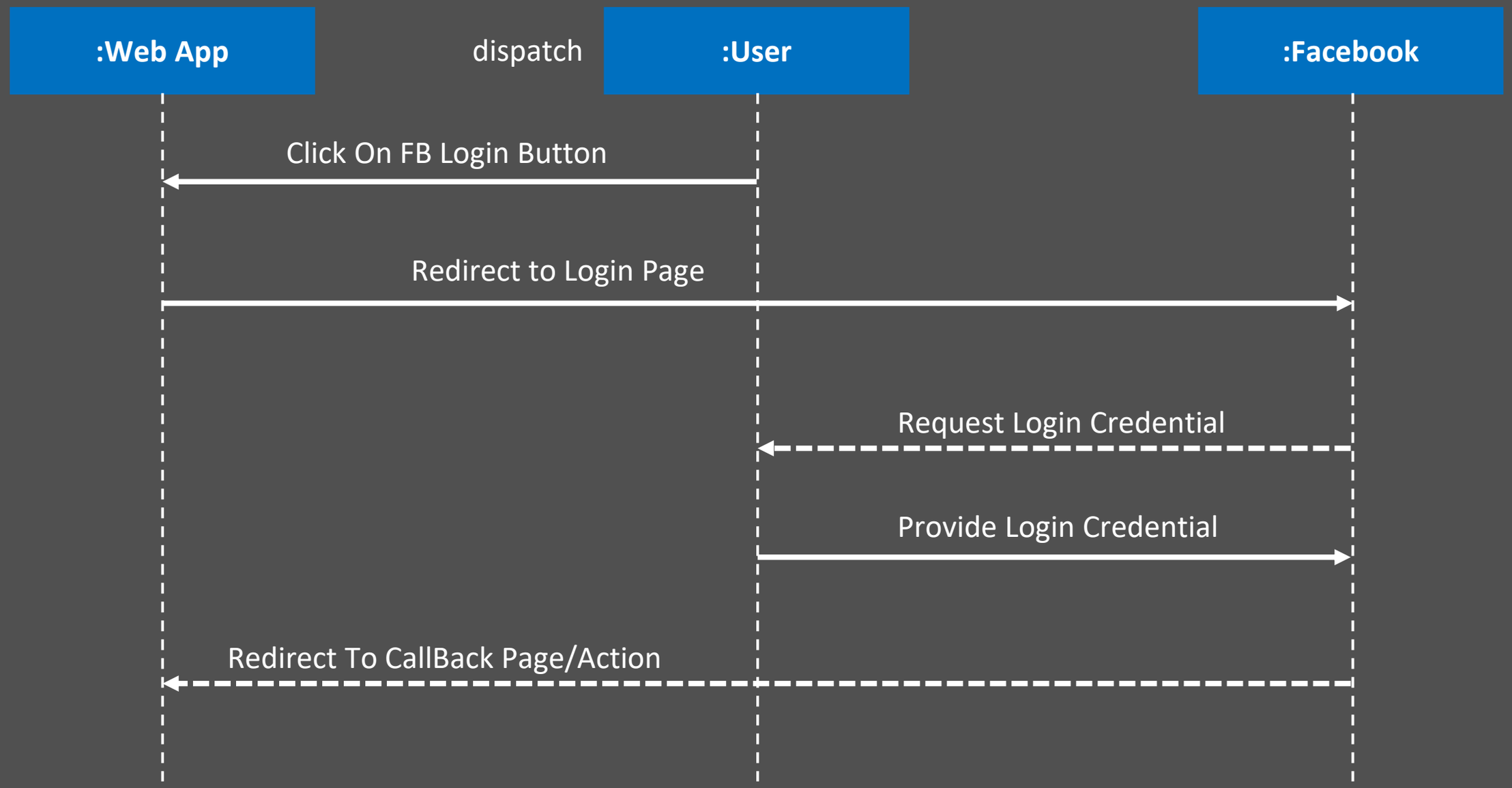
Access Delegation

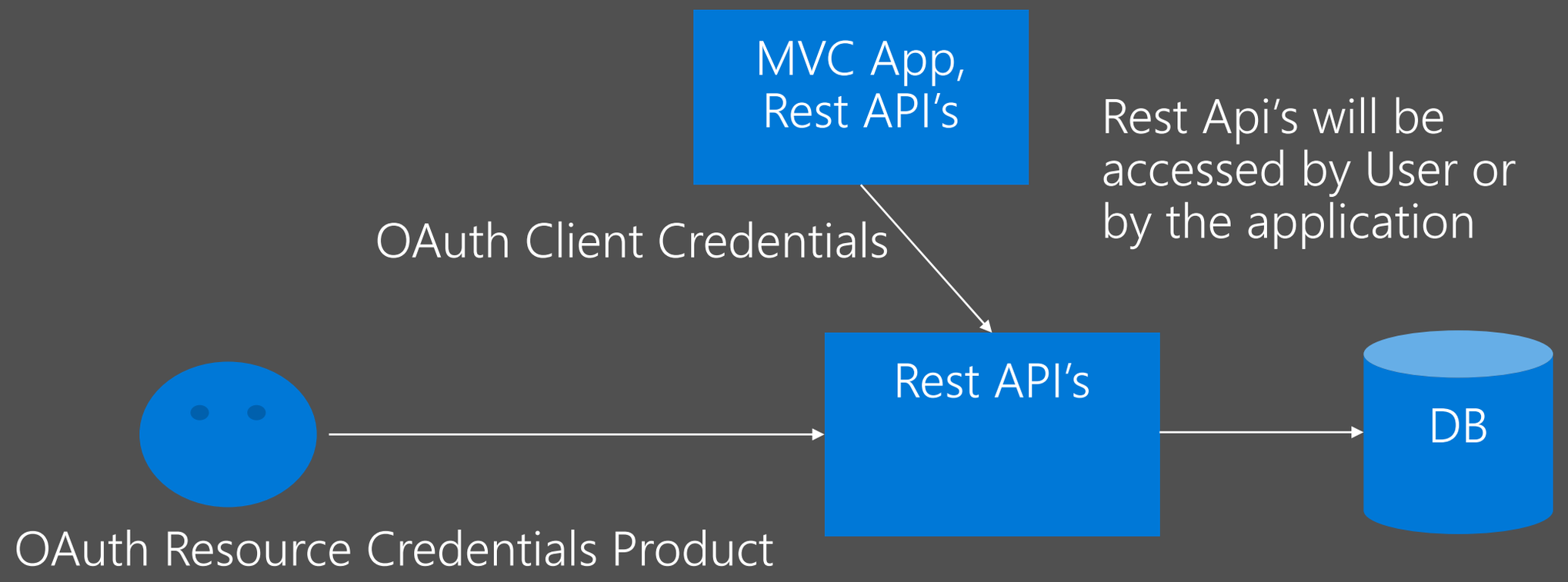
Token Authentication



Oauth

It is Standard protocol for Access Delegation, Facebook,Twitter & Google Support this protocol.


:Web App
dispatch
:User
:Facebook
Click On FB Login Button
Redirect to Login Page
Request Login Credential
Provide Login Credential
Redirect To CallBack Page/Action


MVC App,
Rest API's
Rest Api's will be accessed by User or by the application
OAuth Client Credentials
Rest API's
DB
OAuth Resource Credentials Product

Heading

Content A

Content B

Heading Image