이 발표 자료에는 네이버에서 제공한 나눔 고딕 글꼴이 적용되어 있습니다.
Photo by FadiXd on Unsplash

.NET Conf 2022 x Seoul

풀 스택과 사랑에 빠질 준비, 되셨나요?

.Net Dev

Go/Spring/Node 개발자가 본 .NET6

이은찬

발표 목적

- .NET 이 얼마나 좋은지
- 일반적인 웹 프로그램

일반적인 웹 프로그램이란?

- DB 입출력(DB CRUD)
- 데이터 검증(Validation)
- 인허가(Authentication)
- 캐시(Cache)
- API 문서화(API Documentation)
- 워커(Worker)
- 설정(Config)

예시 : 당근마켓

- DB 입출력(DB CRUD)
- 데이터 검증(Validation)
- 인허가(Authentication)
- 캐시(Cache)
- API 문서화(API Documentation)
- 워커(Worker)
- 설정(Config)

- 중고거래/동네생활 게시글 작성
- 가격 음수 제한, 끌올 제한
- 본인 게시글만 수정 가능
- 게시글 조회 캐시
- Swagger
- 푸시 알람, 신고 제재 등
- 점검 등 기능 플래그(on/off)

1. DB 입출력(DB CRUD)

- DB 입출력 3가지 방법
  - 쌩쿼리 (Raw SQL) -> 오타/타입에러 발생
  - ORM -> 복잡한 비즈니스 하기 힘듦
  - LINQ -> 친절한 Query Builder

```
func (c *ArticlesIteratorCursor) nextRows() (*sqlx.Rows, error) {
        q := `
select
        a.id,
        (CASE WHEN a.mission_id IS NOT NULL THEN (SELECT title FROM missions where id = A.mission_id) ELSE A.title END) as title,
        a.content,
        COALESCE(
                (CASE WHEN a.type!='StoryArticle' THEN NULL
                        ELSE (SELECT array_agg(c.content) FROM comments c WHERE c.article_id = a.id AND c.content IS NOT NULL) END),
                '{}') as comments,
        a.user_id,
        a.region_id,
        a.category_id,
        a.interest_id,
        a.type,
        a.status,
        a.visible,
        a.publish_range,
        CAST(a.price as bigint) as price,
        a.score,
        a.published_at,
        a.created_at,
        a.destroyed_at,
    a.chat_rooms_count,
        a.user_chat_rooms_count,
    a.smb_chat_rooms_count,
        a.first_image_id,
        a.not_published_region_ids,
        a.updated_at,
    a.group_id
from
        articles a
`
```

```
// filter
if (!_.isNil(user)) {
    const filter = await this.getArticleFilter( input: {userId: user.id.toString()})

    if (filter.excludeCarTypes.length > 0)
        baseQuery.andWhere( where: "article.carType NOT IN (:...carTypes)",  parameters: {carTypes: filter.excludeCarTypes})

    if (filter.excludeManufacturers.length > 0)
        baseQuery.andWhere( where: "article.manufacturer NOT IN (:...manufacturer)",  parameters: {manufacturer: filter.excludeManufacturers})

    if (!_.isNil(filter.includeCarYearMin))
        baseQuery.andWhere( where: "article.carYear >= :includeCarYearMin",  parameters: {includeCarYearMin: filter.includeCarYearMin})

    if (!_.isNil(filter.includeCarYearMax))
        baseQuery.andWhere( where: "article.carYear <= :includeCarYearMax",  parameters: {includeCarYearMax: filter.includeCarYearMax})

    if (!_.isNil(filter.includeDriveDistanceMin))
        baseQuery.andWhere( where: "article.driveDistence >= :includeDriveDistanceMin",  parameters: {includeDriveDistanceMin: filter.includeDr

    if (!_.isNil(filter.includeDriveDistanceMax))
        baseQuery.andWhere( where: "article.driveDistence <= :includeDriveDistanceMax",  parameters: {includeDriveDistanceMax: filter.includeDr

    if (!_.isNil(filter.includePriceMin))
        baseQuery.andWhere( where: "article.price >= :includePriceMin",  parameters: {includePriceMin: filter.includePriceMin})

    if (!_.isNil(filter.includePriceMax))
        baseQuery.andWhere( where: "article.price <= :includePriceMax",  parameters: {includePriceMax: filter.includePriceMax})

    // distance km 필터 정보가 있으면 거래 범위 필터 적용
    if (!_.isNil(filter.distanceKm)) {
        const coord = await this.regionCenterCoordById.load(articleQuery.regionId)
        if (coord) {
            const pointForDatabase = pointToDatabase( applicationPoint: {
                lat: coord.lat.toString(),
                lon: coord.pb_long.toString(),
            })

            baseQuery.andWhere(
                where: `ST_DISTANCE(article.coordinate, ST_GeomFromText("POINT(${pointForDatabase.lon} ${pointForDatabase.lat})", 3857)) <= 
                parameters: {
                    distanceM: filter.distanceKm * 1000,
                },
            )
        }
        isRegionConditionAdded = true
    }
}
```

```sql
select count(id)
from article
where deleted_at is null
  and is_hide = 0
  and status != 3
  and is_visible
  and published_at >= DATE_SUB(now(), INTERVAL 60 DAY)
;
```

```fun
@Repository
interface ArticleRepository: CoroutineSortingRepository<Article, Long> {
  fun findAllByDeletedAtAndIsHideAndStatusAndIsVisibleAndPublishedAt(
      deletedAt: DateTime?,
        isHide: Int,
        status: Int,
        isVisible: bool,
        publishedAt: DateTime,
    ): Flow<Article>
}
```

```fun
@Repository
interface ArticleRepository: CoroutineSortingRepository<Article, Long> {
    @Query("""
        select count(id)
        from Article
        where deleted_at is :deletedAt
          and is_hide = :isHide
          and status != :status
          and is_visible = :isVisible
          and published_at >= DATE_SUB(now(), INTERVAL 60 DAY)
    """)
  fun findArticlesPublishedWithin60Days(
      deletedAt: DateTime?,
        isHide: Int,
        status: Int,
        isVisible: bool,
    ): Flow<Article>
}
```

```dotnet
var date60daysAgo = DateTime.UtcNow.AddDays(-60);
var liveArticles = (
    from article in Article
    where
        article.DeletedAt == null
        && article.IsHide == 0
        && article.Status != 3
        && article.IsVisible
        && article.PublishedAt >= date60daysAgo
    select article
).ToList();
```

```
mongoDb.Find(x => x.Pets.Any(p => p.Name == "Fluffy" && p.Age > 21);
```

```
{ Pets: { $elemMatch: { Name: 'Fluffy', Age: { $gt: 21 } } } }
```

Example::MySQL
Example::Mongo

2. 데이터 검증(Validation)

- 강타입 언어
- Data Annotations (Standard)
- Custom Validation

```
str = this.buffer.toString('ascii', start, end);
if (num.toString() === str) {
  return num;
}
return str;
```

https://github.com/sidorares/node-mysql2/blob/f0812c4b34df2e9939592a6e1ee95345696a17c9/lib/packets/packet.js#L482-L487

System.ComponentModel.DataAnnotations Namespace

Reference

Is this page helpful?

Provides attribute classes that are used to define metadata for ASP.NET MVC and ASP.NET data controls.

Classes

AssociatedMetadataTypeTypeDescriptionProvider	Extends the metadata information for a class by adding attributes and property information that is defined in an associated class.
AssociationAttribute	Specifies that an entity member represents a data relationship, such as a foreign key relationship.
CompareAttribute	Provides an attribute that compares two properties.
ConcurrencyCheckAttribute	Specifies that a property participates in optimistic concurrency checks.
CreditCardAttribute	Specifies that a data field value is a credit card number.
CustomValidationAttribute	Specifies a custom validation method that is used to validate a property or class instance.
DataTypeAttribute	Specifies the name of an additional type to associate with a data field.
DisplayAttribute	Provides a general-purpose attribute that lets you specify localizable strings for types and members of entity partial classes.
DisplayColumnAttribute	Specifies the column that is displayed in the referred table as a foreign-key column.
DisplayFormatAttribute	Specifies how data fields are displayed and formatted by ASP.NET Dynamic Data.
EditableAttribute	Indicates whether a data field is editable.
EmailAddressAttribute	Validates an email address.
EnumDataTypeAttribute	Enables a .NET enumeration to be mapped to a data column.
FileExtensionsAttribute	Validates file name extensions.
FilterUIHintAttribute	Represents an attribute that is used to specify the filtering behavior for a column.
KeyAttribute	Denotes one or more properties that uniquely identify an entity.
MaxLengthAttribute	Specifies the maximum length of array or string data allowed in a property.
MetadataTypeAttribute	Specifies the metadata class to associate with a data model class.
MinLengthAttribute	Specifies the minimum length of array or string data allowed in a property.
PhoneAttribute	Specifies that a data field value is a well-formed phone number.
RangeAttribute	Specifies the numeric range constraints for the value of a data field.
RegularExpressionAttribute	Specifies that a data field value in ASP.NET Dynamic Data must match the specified regular expression.
RequiredAttribute	Specifies that a data field value is required.
ScaffoldColumnAttribute	Specifies whether a class or data column uses scaffolding.
StringLengthAttribute	Specifies the minimum and maximum length of characters that are allowed in a data field.
TimestampAttribute	Specifies the data type of the column as a row version.
UIHintAttribute	Specifies the template or user control that Dynamic Data uses to display a data field.
UrlAttribute	Provides URL validation.
ValidationAttribute	Serves as the base class for all validation attributes.
ValidationContext	Describes the context in which a validation check is performed.
ValidationException	Represents the exception that occurs during validation of a data field when the ValidationAttribute class is used.
ValidationResult	Represents a container for the results of a validation request.
Validator	Defines a helper class that can be used to validate objects, properties, and methods when it is included in their associated ValidationAttribute attributes.

https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations?view=net-6.0

Example::CustomValidation

3. 인허가(Authentication)

- 인가 Authentication
- 허가 Authorization

Example::Authentication

4. 캐시(Cache)

- In-Memory Cache
- Redis Cache

Example::Cache

5. API 문서화(API Documentation)

- Swagger

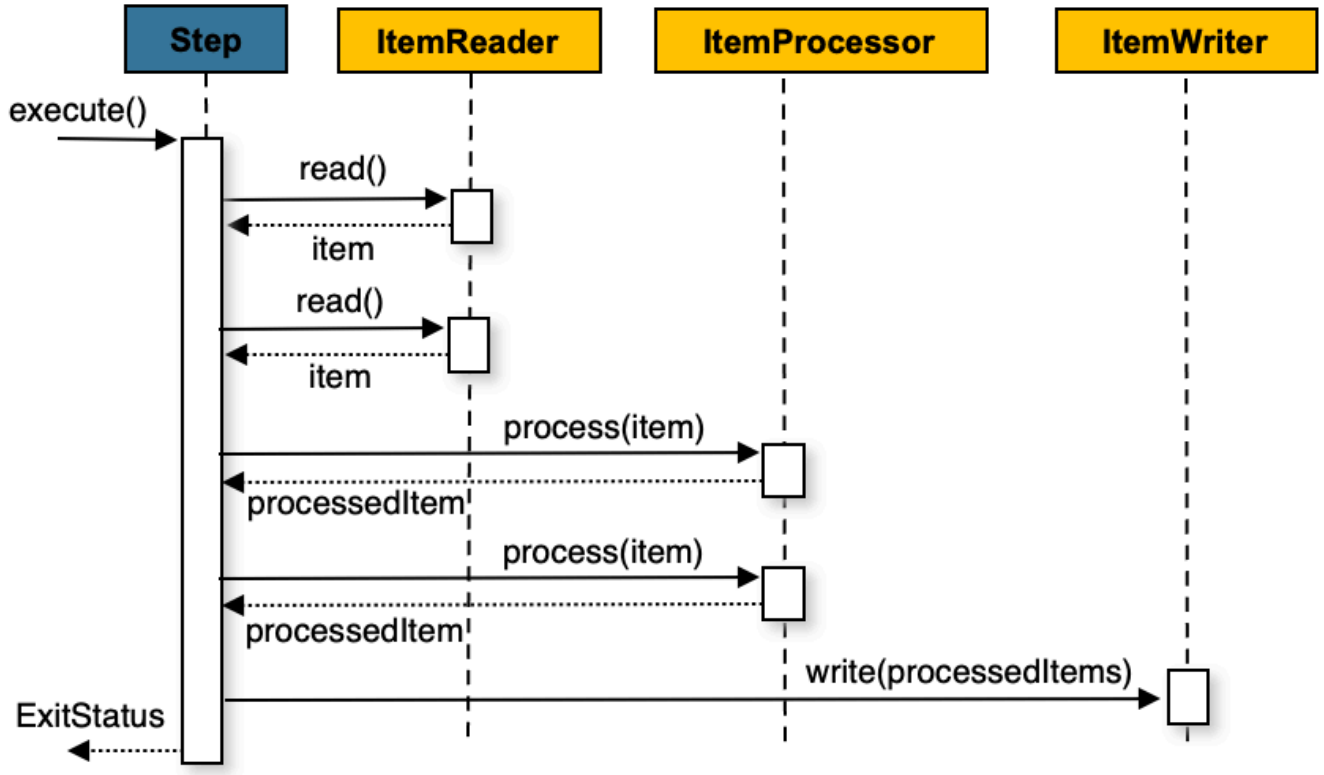
```
@ApiProperty( options: {
    description: "가격(원단위)",
    required: true,
    example: "10000000",
    type: String,
})
@IsNumberString()
price!: string;

@ApiProperty( options: {
    description: "연료 종류",
    required: true,
    example: "gasoline",
    enum: FUEL_TYPE,
})
@IsIn(FUEL_TYPE)
fuelType!: FuelType;

@ApiProperty( options: {
    description: "변속기",
    required: true,
    example: "auto",
    enum: GEAR,
})
@IsIn(GEAR)
gear!: Gear;

@ApiProperty( options: {
    type: SalesArticleOptionDto,
    isArray: true,
    required: false,
})
@Type( typeFunction: () => SalesArticleOptionDto)
@ValidateNested( validationOptions: {
    each: true,
})
options: SalesArticleOptionDto[] = [];

@ApiProperty( options: {
    description: "기타옵션",
    required: true,
    example: null,
})
@IsOptional()
@IsString()
etcOption?: string;
```

```
/// <summary>
/// dto 설명
/// </summary>
1 usage   new *
class Dto
{
    /// <summary>
    /// 가격(원 단위)
    /// </summary>
    public uint Price { get; set; }
    /// <summary>
    /// 연료 종류
    /// </summary>
    public FuelType FuelType { get; set; }
    /// <summary>
    /// 기어
    /// </summary>
    public Gear Gear { get; set; }
}

[JsonConverter(typeof(JsonStringEnumConverter))]
1 usage   Eunchan Lee   9 exposing APIs
enum FuelType
{
    Gasoline,
    Diesel,
    Lpg,
    Cng,
    Electronic,
    FuelCellElectronic,
    Hybrid,
}

[JsonConverter(typeof(JsonStringEnumConverter))]
1 usage   Eunchan Lee   4 exposing APIs
enum Gear
{
    Auto,
    Manual,
}
```

Example::MinimalAPI

6. 워커(Worker)

- One Time
- Cron
- Queue
- Parallel



https://docs.spring.io/spring-batch/docs/current/reference/html/step.html

Example::Worker

7. 설정(Config)

- IOptions
- IOptionsSnapshot
- IOptionsMonitor

Example::Config

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participatio
- MSDN

```
var show = new Command( name: "show",  description: "show my ip");
show.AddOption(new Option<bool>( alias: "--check",  description: "option description"));
show.AddOption(new Option<int>( alias: "--delay",  description: "delay in ms"));
show.AddOption(new Option<Env>( alias: "--env",  description: "alpha, prod"));
show.Handler = CommandHandler.Create(async (bool check, int delay, Env env) =>
{
```

```
program
  .command( nameAndArgs: 'log')
  .requiredOption( flags: '--service <service name>',  description: 'realty-server')
  .requiredOption( flags: '--env <env>',  description: 'alpha prod')
  .option( flags: '--cluster <name>',  description: 'alpha-kr-b prod-kr-a')
  .option( flags: '--timestamp')
  .action( fn: async (args: {
        env: ENV,
        service: string,
        cluster?: string,
        timestamp?: boolean,
    }) => {
    if (!(args.service in Services)) throw Error(`${args.service} 서비스를 찾을 수 없어요`);
    const service = Services[args.service];
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
drwxr-xr-x    4 danny  staff   128B Dec 18 01:39 ./
drwxr-xr-x  231 danny  staff   7.2K Dec 18 01:26 ../
-rwxr-xr-x    1 danny  staff    21M Dec 18 01:39 dgn2*
-rw-r--r--    1 danny  staff    15K Dec 18 01:39 dgn2.pdb
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
[Fact]
usages  overrides  Eunchan Lee  ext methods  exposing APIs
public void SearchKyobo()
{
    using var client = _serverFixture.CreateClient();
    var res :GetResponse = client.GetFromJsonAsync<AddressHandler.GetResponse>( requestUri: "/api/v1/address?query=교보&page=1").Result!;

    var json :string = JsonSerializer.Serialize(res);
    _testOutputHelper.WriteLine(json);
    Assert.Contains( expectedSubstring: "465, Gangnam-daero, Seocho-gu, Seoul", actualString: json);
}
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
// db
builder.Services.AddDbContext<MysqlContext>();
// service
builder.Services.AddScoped<OptionBugFixChatService>();
builder.Services.AddScoped<AddressService>();
builder.Services.AddScoped<ShortMultiPromotionChatService>();
// grpc
builder.Services.AddScoped<RocketChatService>();
builder.Services.AddScoped<BigPictureService>();
builder.Services.AddScoped<KarrotAuthService>();
```

```
app.MapGet( pattern: "/healthcheck",
    handler: (
        HttpContext ctx,
        IHostEnvironment env,
        MysqlContext mysqlContext,
        KarrotAuthService karrotAuthService,
        BigPictureService bigPictureService
    ) =>
    {
```

이 외 .NET 의 장점

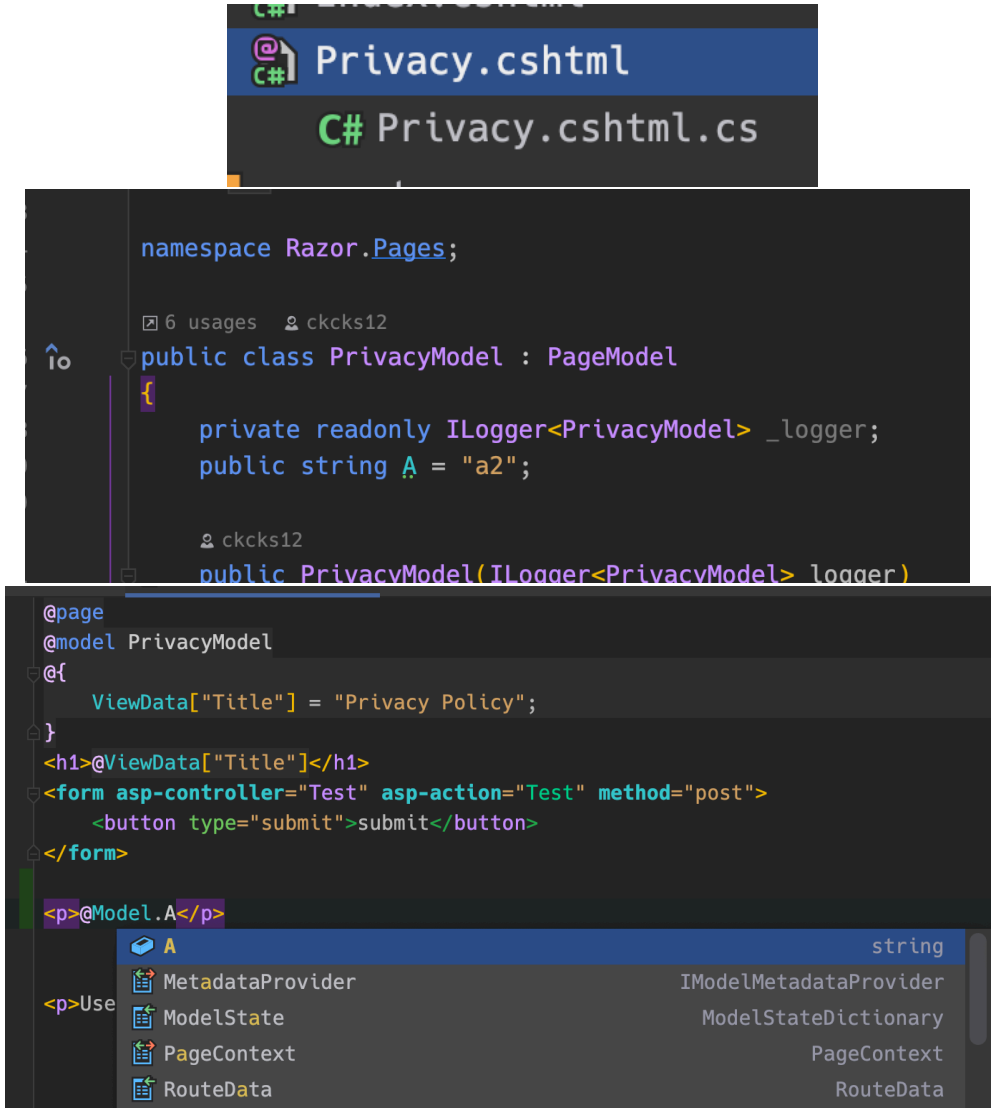
- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
<!-- grpc -->
<ItemGroup>
    <None Include="Protos/pbregistry/protos/**" />
    <Protobuf Include="Protos/pbregistry/protos/rocket-chat/proto/rpc/v2/*.proto" GrpcSevices="Client" ProtoRoot="Protos/pbregistry/prot
    <Protobuf Include="Protos/pbregistry/protos/big-picture/rpc/v2/bigpicture.proto" GrpcServices="Client" ProtoRoot="Protos/pbregistry/
    <Protobuf Include="Protos/pbregistry/protos/google/**/*.proto" GrpcServices="None" ProtoRoot="Protos/pbregistry/protos" />
    <Protobuf Include="Protos/pbregistry/protos/karrotauth/rpc/v1/*.proto" GrpcServices="Client" ProtoRoot="Protos/pbregistry/protos/kar
    <Protobuf Include="Protos/pbregistry/protos/karrotauth/rpc/*.proto" GrpcServices="Client" ProtoRoot="Protos/pbregistry/protos" />
</ItemGroup>
<!-- // grpc -->
```

```
Eunchan Lee
public BigPictureService(IOptions<AppConfig> appConfig)
{
    _appConfig = appConfig.Value;

    var ch :GrpcChannel = GrpcChannel.ForAddress(_appConfig.BigPictureGrpcAddress,
    {
        Credentials = ChannelCredentials.Insecure
    });

    _clientV2 = new ImageSvc.ImageSvcClient(ch);
}
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN



이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
using System.Text.Json;
using Flurl;
using Flurl.Http;

var res :dynamic? = await "https://httpbin.org/anything"// string
    .SetQueryParams( values: new
    {
        query = "query"
    })// Url
    .SendJsonAsync(HttpMethod.Post, data: new
    {
        body = "body"
    })// Task<IFlurlResponse>
    .ReceiveJson();// Task<dynamic>

// json print
Console.WriteLine(JsonSerializer.Serialize(res));
Console.WriteLine($"query : {res.args.query}");
Console.WriteLine($"body : {res.json.body}");
res.
    args                    ???     Field args
    json                    ???
    Equals                  bool
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

```
$ dn watch run
watch : Hot reload enabled. For a list of supported edits, see https://aka.ms/dotnet/hot-reload. Press "Ctrl + R" to restart.
watch : Building...
  Determining projects to restore...
  All projects are up-to-date for restore.
  MinimalAPI -> /Users/danny/3unch4n/dotnet-example/example/MinimalAPI/bin/Debug/net6.0/MinimalAPI.dll
watch : Started
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

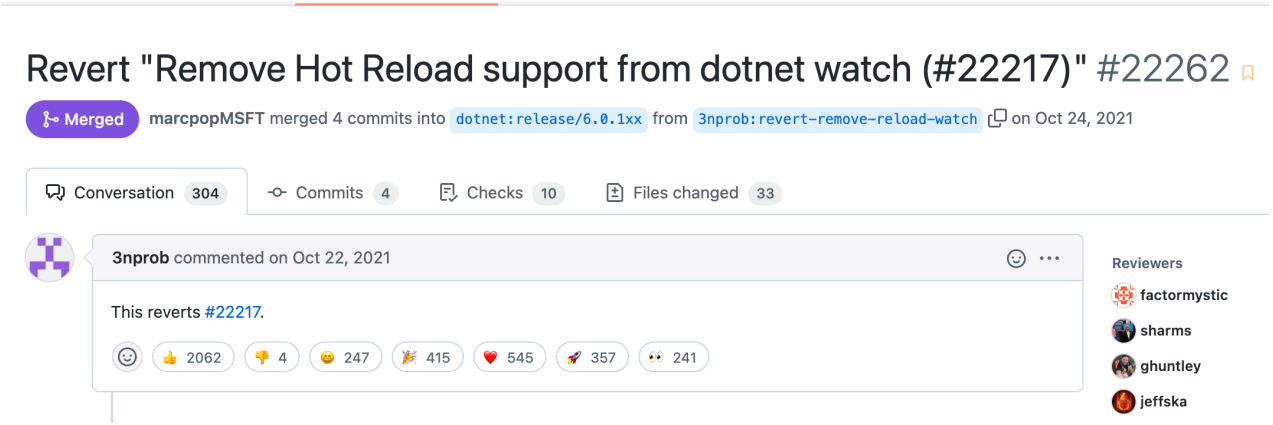
```
IL code
public static class HostingAbstractionsHostExtensions
{
    /// <summary>
    /// Starts the host synchronously.
    /// </summary>
    /// <param name="host">The <see cref="IHost"/> to start.</param>
    IL code
    public static void Start(this IHost host)
    {
        host.StartAsync().GetAwaiter().GetResult();
    }

    /// <summary>
    /// Attempts to gracefully stop the host with the given timeout.
    /// </summary>
    /// <param name="host">The <see cref="IHost"/> to stop.</param>
    /// <param name="timeout">The timeout for stopping gracefully. Once expired the
    /// server may terminate any remaining active connections.</param>
    /// <returns>The <see cref="Task"/> that represents the asynchronous operation.</returns>
    IL code
    public static async Task StopAsync(this IHost host, TimeSpan timeout)
    {
        using CancellationTokenSource cts = new CancellationTokenSource(timeout);
        await host.StopAsync(cts.Token).ConfigureAwait(false);
    }

    /// <summary>
    /// Block the calling thread until shutdown is triggered via Ctrl+C or SIGTERM.
    /// </summary>
    /// <param name="host">The running <see cref="IHost"/>.</param>
    IL code
    public static void WaitForShutdown(this IHost host)
    {
        host.WaitForShutdownAsync().GetAwaiter().GetResult();
    }

    /// <summary>
    /// Runs an application and block the calling thread until host shutdown.
    /// </summary>
    /// <param name="host">The <see cref="IHost"/> to run.</param>
    IL code
    public static void Run(this IHost host)
    {
        host.RunAsync().GetAwaiter().GetResult();
    }

    /// <summary>
    /// Runs an application and returns a <see cref="Task"/> that only completes when the token is triggered or shutdown is trigge
    /// The <paramref name="host"/> instance is disposed of after running.
    /// </summary>
    /// <param name="host">The <see cref="IHost"/> to run.</param>
    /// <param name="token">The token to trigger shutdown.</param>
    /// <returns>The <see cref="Task"/> that represents the asynchronous operation.</returns>
    IL code
    public static async Task RunAsync(this IHost host, CancellationToken token = default)
    {
        try
```

이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN

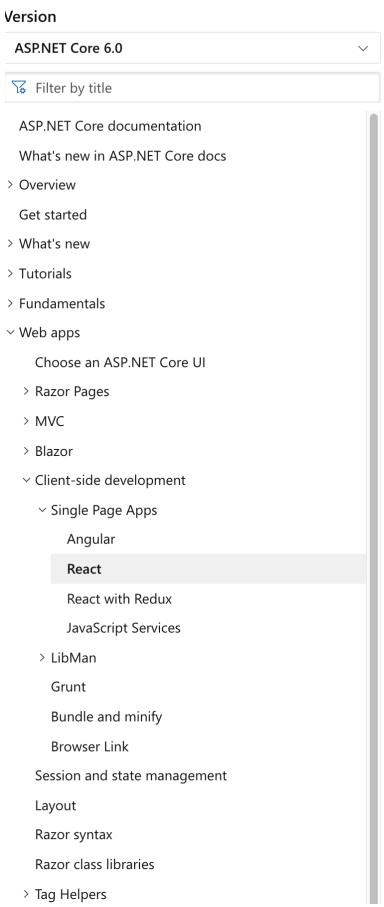
https://devblogs.microsoft.com/dotnet/net-hot-reload-support-via-cli/

https://github.com/dotnet/sdk/pull/22262



이 외 .NET 의 장점

- Command Line App
- Single Executable
- Testable
- DI
- gRPC
- MVC Model
- HttpParser (dynamic)
- Hot Reload
- Internal Source Readability
- Github Community Participation
- MSDN



Use the React project template with ASP.NET Core

Article • 12/28/2021 • 4 minutes to read • +4

Is this page helpful?

The updated React project template provides a convenient starting point for ASP.NET Core apps using React and create-react-app (CRA) conventions to implement a rich, client-side user interface (UI).

The template is equivalent to creating both an ASP.NET Core project to act as an API backend, and a standard CRA React project to act as a UI, but with the convenience of hosting both in a single app project that can be built and published as a single unit.

The React project template isn't meant for server-side rendering (SSR). For SSR with React and Node.js, consider Next.js or Razzle.

Create a new app

If you have ASP.NET Core 2.1 or later installed, there's no need to install the React project template.

Create a new project from a command prompt using the command `dotnet new react` in an empty directory. For example, the following commands create the app in a *my-new-app* directory and switch to that directory:

.NET CLI Copy

```
dotnet new react -o my-new-app
cd my-new-app
```

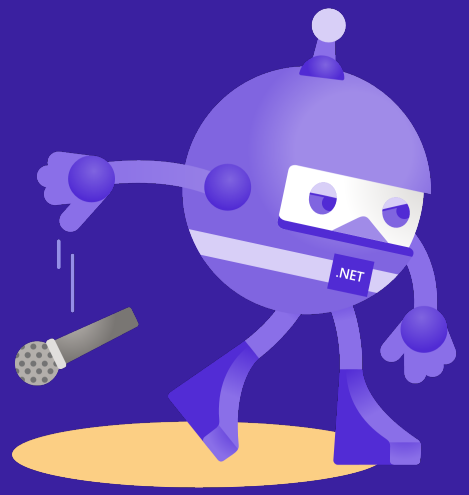
Run the app from either Visual Studio or the .NET Core CLI:

Visual Studio | .NET Core CLI

Open the generated *.csproj* file, and run the app as normal from there.

The build process restores npm dependencies on the first run, which can take several minutes. Subsequent builds are much faster.

고맙습니다!



두 칸 슬라이드

데모

간단한 설명

샘플 코드

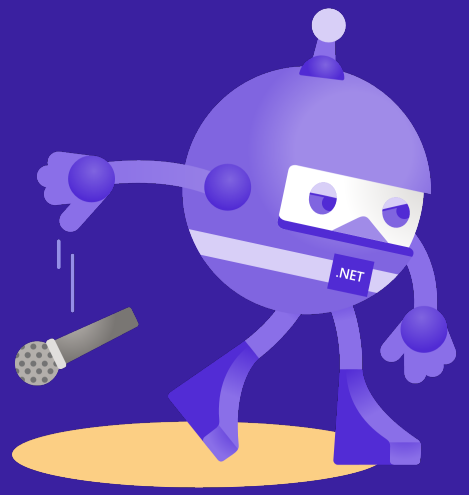
샘플 코드

RELEASED

발표

- 강조할 내용 1
- 강조할 내용 2
- 강조할 내용 3

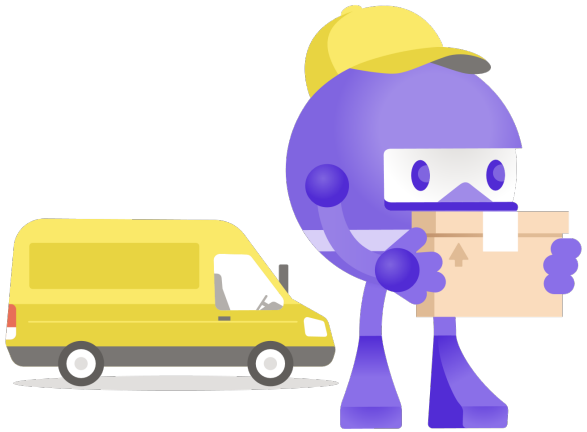
고맙습니다!

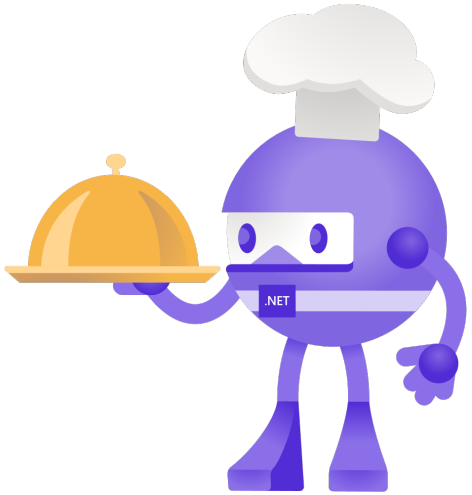


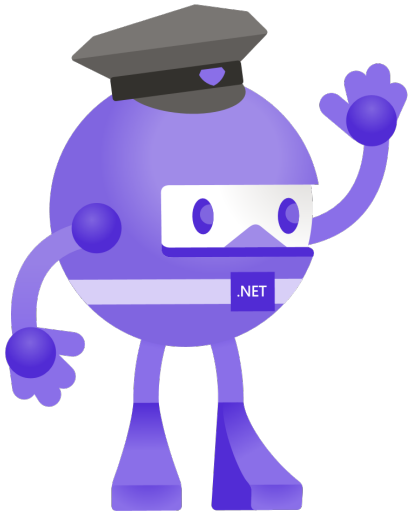
RELEASED

목차

- 일반적인 웹 프로그램을 위한 .NET
- 개발 경험 향상 위한 .NET

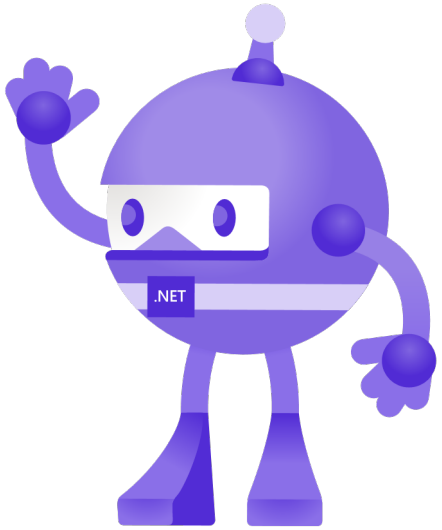



.NET

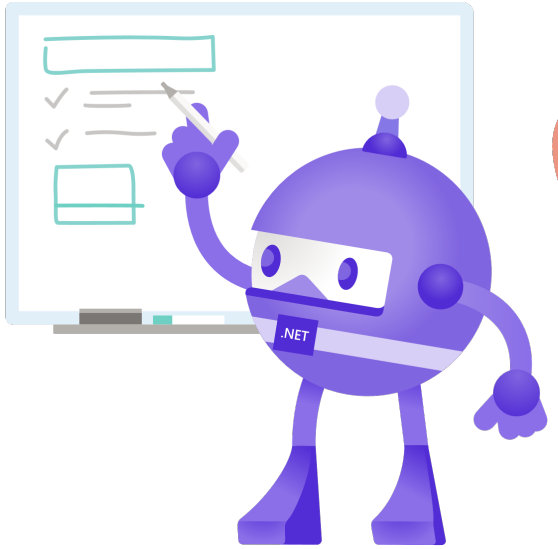

.NET

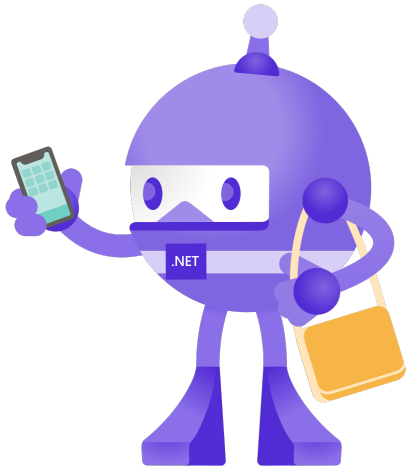
.NET

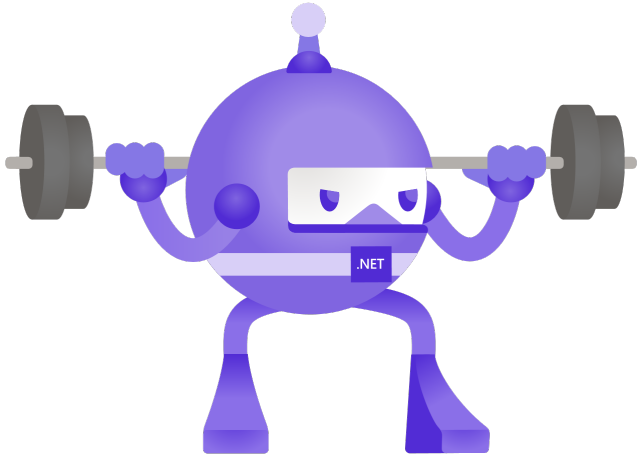
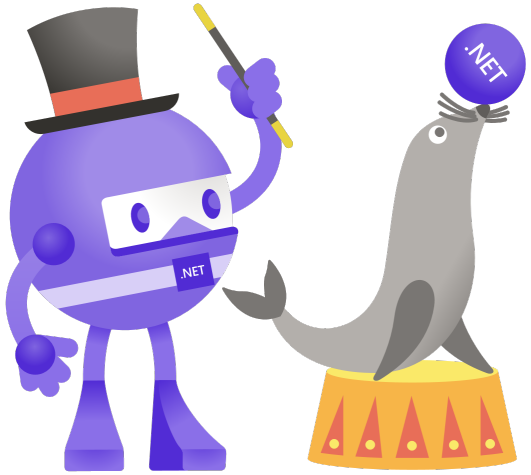
.NET

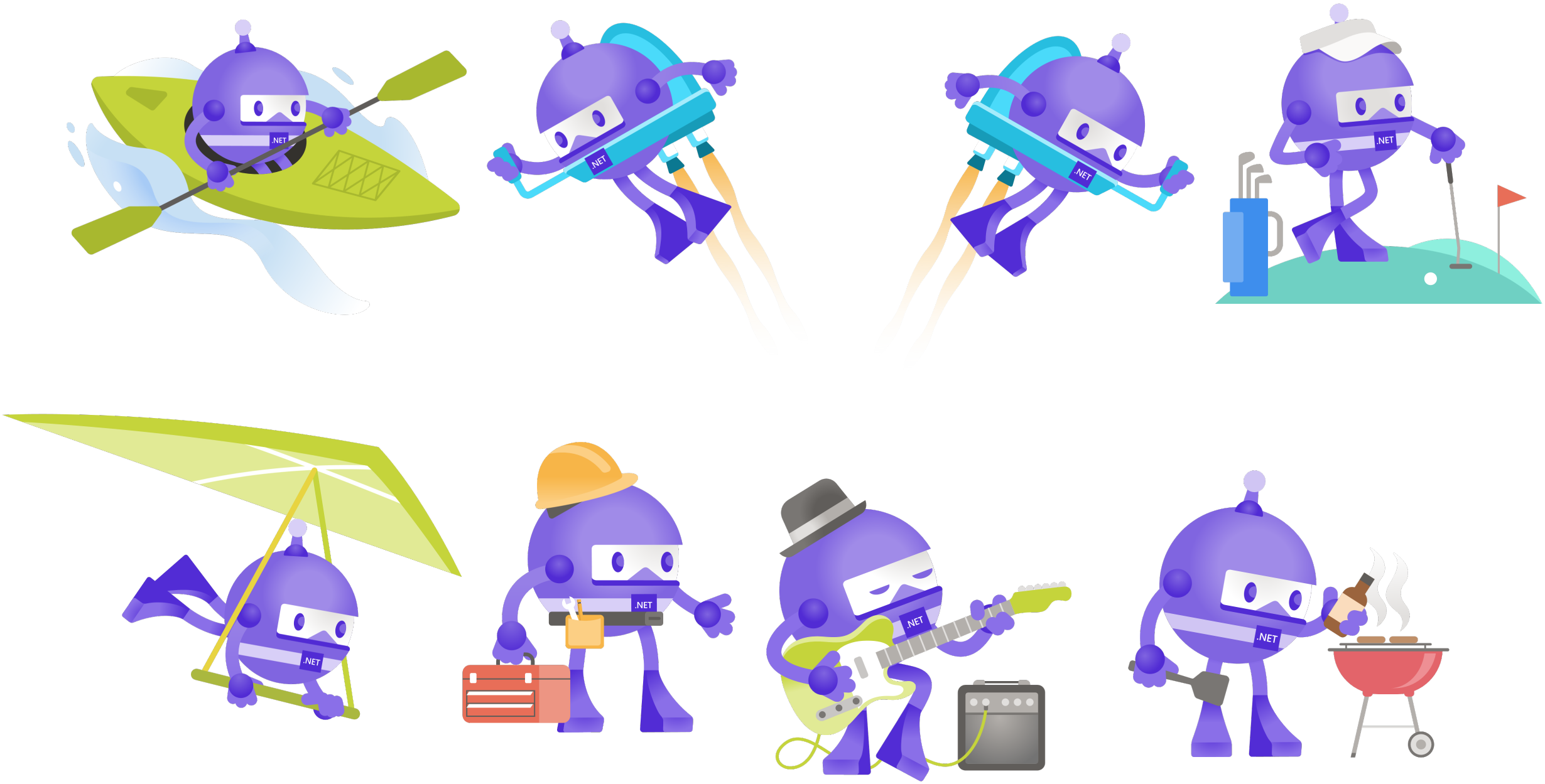

.NET

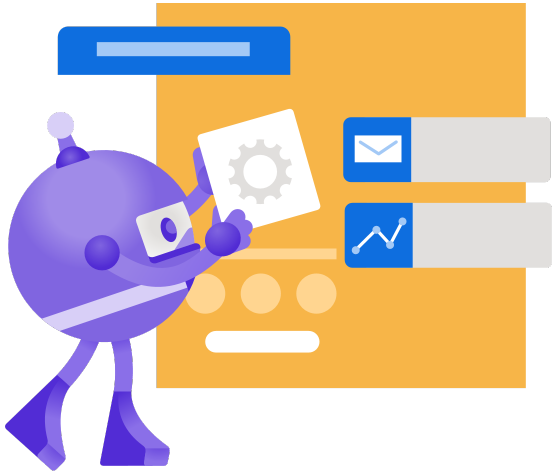
.NET

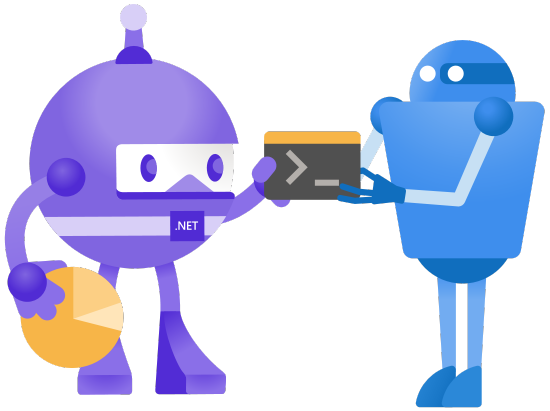

.NET

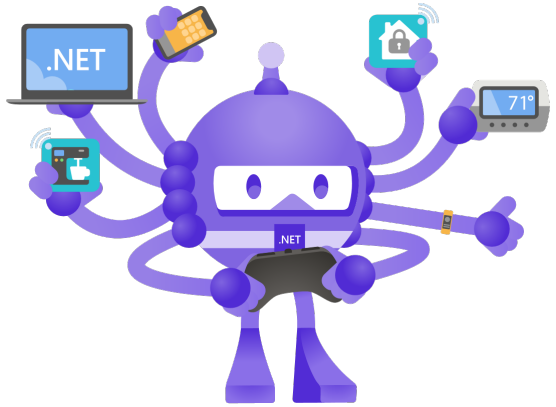
.NET

.NET

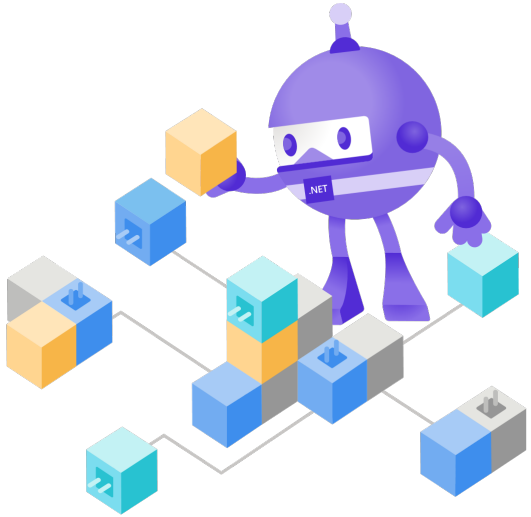
.NET

.NET

.NET
.NET

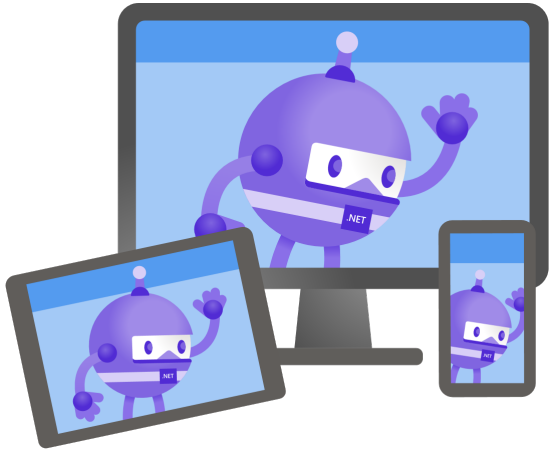

.NET
.NET
.NET
.NET
.NET
.NET
.NET
.NET

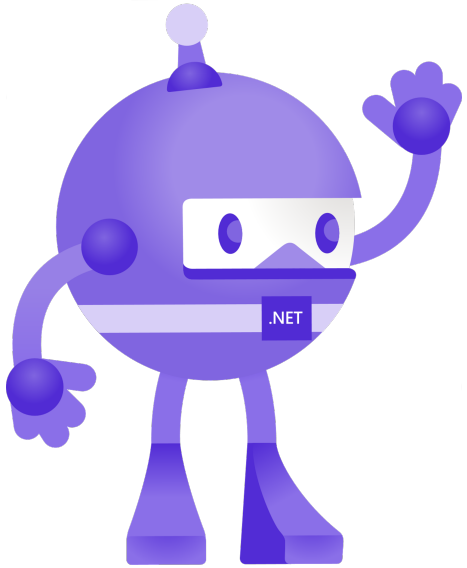





.NET
71°

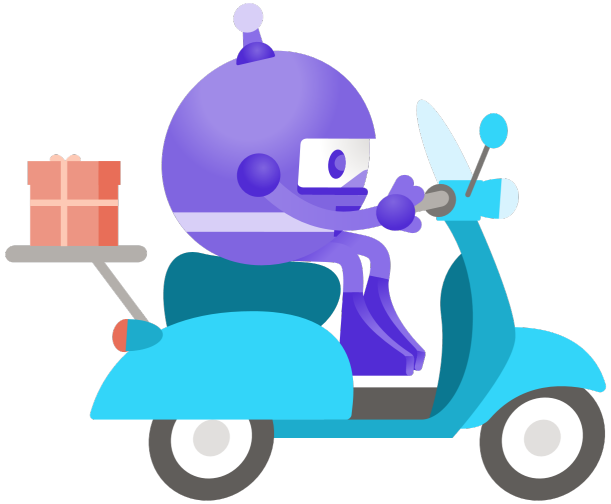





.NET

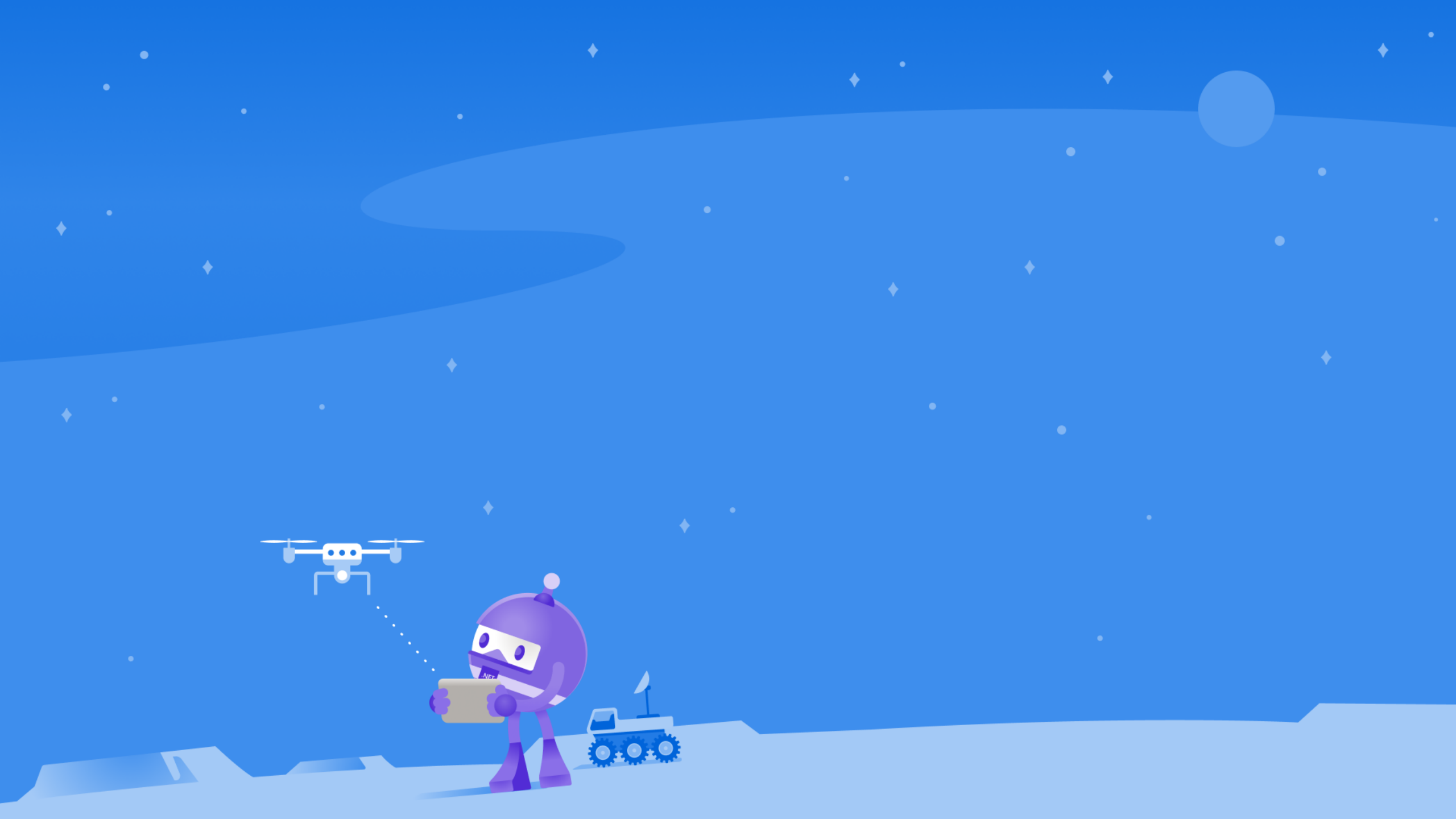
.NET

.NET



.NET

.NET


.NET