

در این مطلب می‌خواهم روش استفاده از Async&Await رو براتون بگم. Async&Await خط و مشی جدید Microsoft برای تولید متدهای Async هستش که نوشتن این متدها رو خیلی جذاب کرده و کاربردهای خیلی زیادی هم داره. مثلاً هنگام استفاده از Web Api در برنامه‌های تحت ویندوز نظیر WPF این روش خیلی به ما کمک می‌کنه و در کل نوشتن Parallel Programming را خیلی جالب کرده.

برای اینکه بتونم قدرت و راحتی کار با این ابزار رو به خوبی نشون بدم ابتدا یک مثال رو به روشی قدیمی‌تر پیاده سازی می‌کنم. بعد پیاده سازی همین مثال رو به روش جدید بهتون نشون می‌دم.

می‌خواهم یک برنامه بنویسم که لیستی از محصولات رو به صورت Async در خروجی چاپ کنه. ابتدا کلاس مدل:

```
public class Product
{
    public int Id { get; set; }

    public string Name { get; set; }
}
```

حالا کلاس ProductService رو می‌نویسم:

```
public class ProductService
{
    public ProductService()
    {
        ListOfProducts = new List<Product>();
    }

    public List<Product> ListOfProducts
    {
        get;
        private set;
    }

    private void InitializeList( int toExclusive )
    {
        Parallel.For( 0 , toExclusive , ( int counter ) =>
        {
            ListOfProducts.Add( new Product()
            {
                Id = counter ,
                Name = "DefaultName" + counter.ToString()
            } );
        } );
    }

    public IAsyncResult BeginGetAll( AsyncCallback callback , object state )
    {
        var myTask = Task.Run<IEnumerable<Product>>( () =>
        {
            InitializeList( 100 );
            return ListOfProducts;
        } );
        return myTask.ContinueWith( x => callback( x ) );
    }

    public IEnumerable<Product> EndGetAll( IAsyncResult result )
    {
        return ( ( Task<IEnumerable<Product>> )result ).Result;
    }
}
```

در کلاس بالا دو متد مهم دارم. متد اول آن BeginGetAll است و همونطور که می‌بینید خروجی اون از نوع IAsyncResult است و باید هنگام استفاده، اونو به متد EndGetAll پاس بدم تا خروجی مورد نظر به دست بیاد. متد InitializeList به تعداد ورودی آیتم به لیست اضافه می‌کند و اونو به Callback می‌فرسته. در نهایت برای اینکه بتونم از این

کلاس ها استفاده کنم باید به صورت زیر عمل بشه:

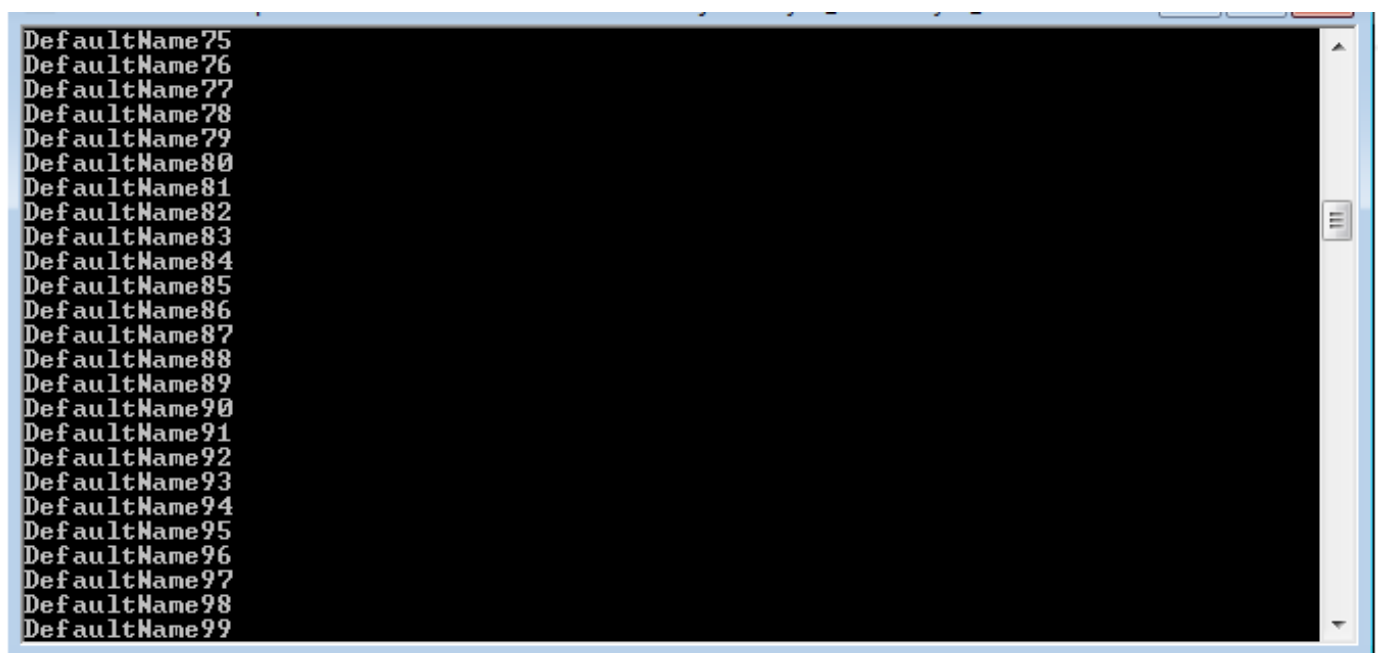
```
class Program
{
    static void Main( string[] args )
    {
        GetAllProducts().ToList().ForEach( ( Product item ) =>
        {
            Console.WriteLine( item.Name );
        } );

        Console.ReadLine();
    }

    public static IEnumerable<Product> GetAllProducts()
    {
        ProductService service = new ProductService();

        var output = Task.Factory.FromAsync<IEnumerable<Product>>( service.BeginGetAll ,
service.EndGetAll , TaskCreationOptions.None );
        return output.Result;
    }
}
```

خیلی راحت بود؛ درسته. خروجی مورد نظر رو می بینید:



```
DefaultName75
DefaultName76
DefaultName77
DefaultName78
DefaultName79
DefaultName80
DefaultName81
DefaultName82
DefaultName83
DefaultName84
DefaultName85
DefaultName86
DefaultName87
DefaultName88
DefaultName89
DefaultName90
DefaultName91
DefaultName92
DefaultName93
DefaultName94
DefaultName95
DefaultName96
DefaultName97
DefaultName98
DefaultName99
```

حالا همین کلاس بالا رو به روش Async&Await می نویسم:

```
public async Task<IEnumerable<Product>> GetAllAsync()
{
    var result = Task.Run( () =>
    {
        InitializeList( 100 );
        return ListOfProducts;
    } );
    return await result;
}
```

در متد بالا به جای استفاده از 2 متد از یک متد GetAllAsync استفاده کردم که خروجی آون از نوع async

Task<IEnumerable<Product>> GetAllProducts() << بود و برای استفاده از اون کافیه در کلاس Program کد زیر رو بنویسم

```
class Program
{
    static void Main( string[] args )
    {
        GetAllProducts().Result.ToList().ForEach( ( Product item ) =>
        {
            Console.WriteLine( item.Name );
        } );

        Console.ReadLine();
    }

    public static async Task<IEnumerable<Product>> GetAllProducts()
    {
        ProductService service = new ProductService();

        return await service.GetAllAsync();
    }
}
```

فکر کنم همتون موافقید که روش Async&Await هم از نظر نوع کد نویسی و هم از نظر راحتی کار خیلی سرتیره. یکی از مزایای مهم این روش اینه که همین مراحل رو می تونید در هنگام استفاده از WCF در پروژه تکرار کنید. به خوبی کار می کنه.