Dotnet Core Best Practices

HTTP Method Seçimi

HTTP (Hypertext Transfer Protocol) protokolü, web üzerinde bilgi alışverişi yapmak için kullanılan bir protokoldür. HTTP, belirli amaçlara hizmet eden bir dizi metod (method) veya eylem içerir. Bu metodlar, HTTP isteğinin amacını belirtir ve isteğin ne tür bir işlem gerçekleştirmesi gerektiğini tanımlar. İşte bazı temel HTTP metodları:

GET:

- **Kaynaklardan bilgi almak için kullanılır.**
- **İstek gövdesi yoktur ve veri göndermez.**
- **Cevap olarak bir kaynak veya durum kodu dönebilir.**

POST:

- **Yeni bir kaynak oluşturmak veya varolan bir kaynağı güncellemek için kullanılır.**
- **İstek gövdesinde veri taşır.**
- **Genellikle form verileri veya JSON gibi veri formatlarıyla kullanılır.**

PUT:

- **Belirtilen URI'deki kaynağı oluşturmak veya güncellemek için kullanılır.**
- **İstek gövdesinde taşınan veri, kaynağın tamamını temsil eder.**

DELETE:

- **Belirtilen URI'deki kaynağı silmek için kullanılır.**
- **İstek gövdesi olmaz veya boştur.**

Http Durum Kodları

1xx - Bilgi:

- **100 Continue: İstemcinin, devam etmesi durumunda bir isteğin tamamlanabileceğini bildiren bir yanıt.**

2xx - Başarılı:

- **200 OK: İstek başarıyla gerçekleştirildi.**
- **201 Created: İstekle yeni bir kaynak oluşturuldu.**

- **204 No Content: İstek başarılı olsa da yanıt içerik taşımaz.**

3xx - Yönlendirme:

- **301 Moved Permanently: Kaynak, kalıcı olarak başka bir URI'ye taşındı.**
- **302 Found: Kaynak, geçici olarak başka bir URI'ye taşındı.**
- **304 Not Modified: Kaynak, istemcinin önbelleklenmiş bir sürümüne dayalı olarak güncellenmedi.**

4xx - İstemci Hatası:

- **400 Bad Request: İstek yapısal olarak yanlış veya anlaşılamaz.**
- **401 Unauthorized: Kimlik doğrulama gerekiyor veya başarısız oldu.**
- **403 Forbidden: İstemci, kaynağa erişim iznine sahip değil.**
- **404 Not Found: Belirtilen URI'deki kaynak bulunamadı.**
- **405 Method Not Allowed: Belirtilen metod, bu kaynağa uygulanamaz.**
- **429 Too Many Requests: İstemci, belirli bir süre içinde çok fazla istek gönderdi.**

5xx - Sunucu Hatası:

- **500 Internal Server Error: Sunucu genel bir hata ile karşılaştı.**
- **501 Not Implemented: Sunucu, isteği yerine getirmek için gerekli yeteneklere sahip değil.**
- **503 Service Unavailable: Sunucu şu anda hizmet veremiyor. Geçici olarak bakım modunda olabilir veya aşırı yük altında olabilir.**

Bu durum kodları, bir HTTP yanıtının genel durumunu belirtir. Ancak, bu kodların her biri daha spesifik durumları ifade edebilir ve belirli durumlar için uygun olan kodlar değişebilir. Bu durum kodlarını doğru bir şekilde anlamak, web uygulamalarının hata ayıklamasını ve hata yönetimini geliştirmek açısından önemlidir.(Şu web sitesini de inceleyebilirsiniz.https://http.cat/)

Doğru Endpoint Kullanımı

GET - Tüm Makaleleri Getir:

- **Açıklama: Tüm makaleleri getiren bir endpoint.**
- **Endpoint: GET /api/articles**

GET - Belirli Bir Makaleyi Getir:

- **Açıklama: Belirli bir makaleyi ID ile getiren bir endpoint.**
- **Endpoint: GET /api/articles/{id}**

POST - Yeni Makale Oluştur:

- **Açıklama: Yeni bir makale oluşturan bir endpoint.**
- **Endpoint: POST /api/articles**

PUT - Makaleyi Güncelle:

- **Açıklama: Belirli bir makaleyi güncelleyen bir endpoint.**
- **Endpoint: PUT /api/articles/{id}**

DELETE - Makaleyi Sil:

- **Açıklama: Belirli bir makaleyi silen bir endpoint.**
- **Endpoint: DELETE /api/articles/{id}**

GET - Makaleye Yorumları Getir:

- **Açıklama: Belirli bir makaleye ait yorumları getiren bir endpoint.**
- **Endpoint: GET /api/articles/{id}/comments**

POST - Yeni Yorum Ekle:

- **Açıklama: Belirli bir makaleye yeni bir yorum ekleyen bir endpoint.**
- **Endpoint: POST /api/articles/{id}/comments**

GET - Kategorilere Göre Makaleleri Getir:

- **Açıklama: Belirli bir kategoriye ait makaleleri getiren bir endpoint.**
- **Endpoint: GET /api/categories/{category}/articles**

GET - Kullanıcının Tüm Makalelerini Getir:

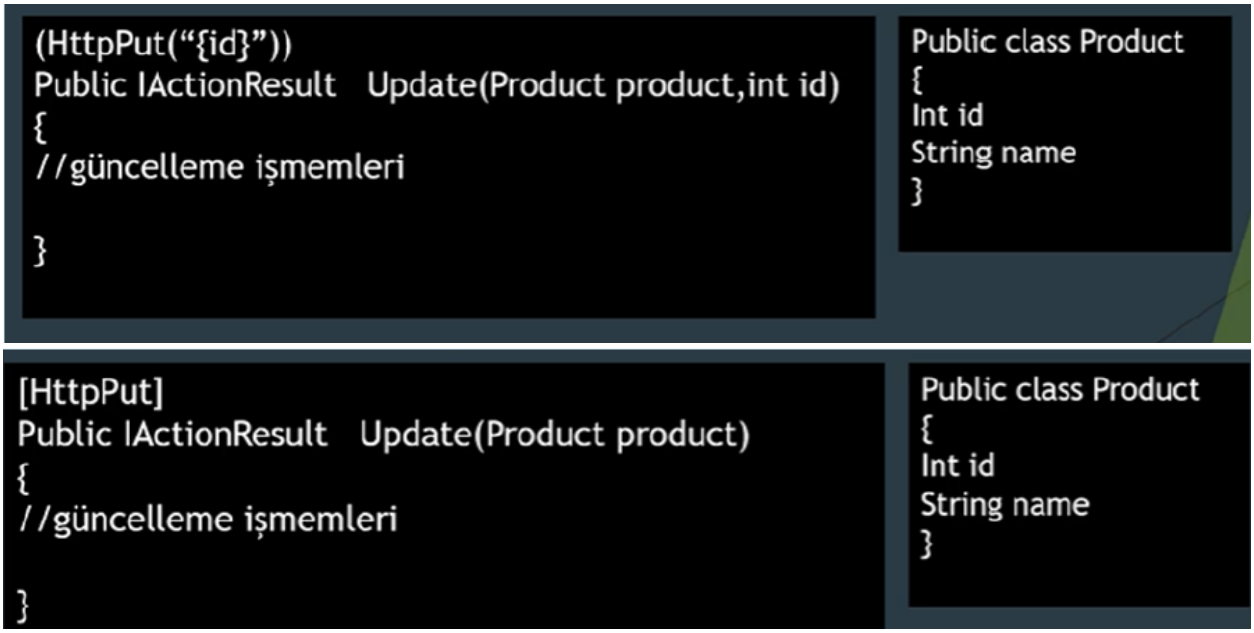
- **Açıklama: Belirli bir kullanıcının tüm makalelerini getiren bir endpoint.**
- **Endpoint: GET /api/users/{userId}/articles**

Bu örneklerde dikkat edilmesi gereken noktalar şunlar:

- **Endpoint isimleri açık ve anlaşılır.**

- **HTTP metodları, istenen operasyona uygun olarak kullanılmış.**
- **Path parametreleri (örneğin {id}), belirli bir kaynağa işaret etmek için kullanılmış.**
- **Endpoint'lerde CORS kontrolleri ve güvenlik önlemleri düşünülmüş.**
- **Query parametreleri (örneğin category) ile filtreleme sağlanmış.**
- **Versiyonlama için özel bir endpoint eklenmemiş, ancak bu büyük projelerde gerekebilir.**

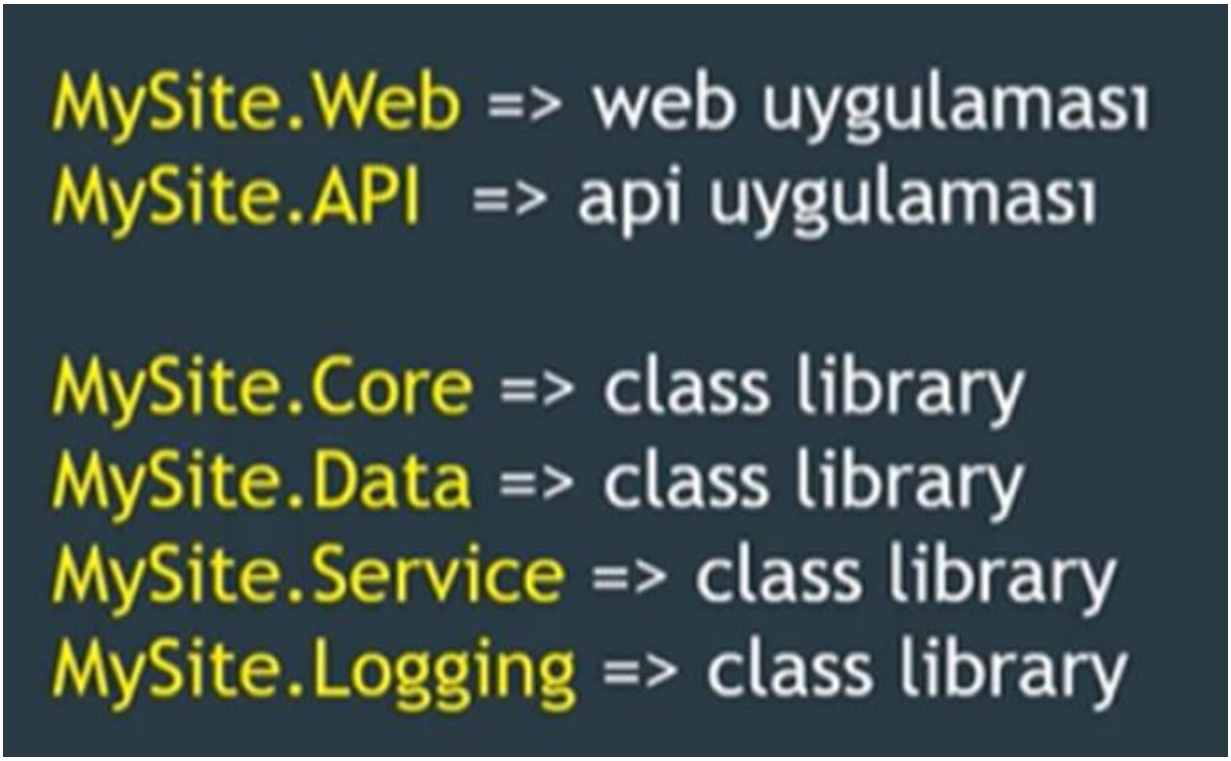
REQUEST İÇERİSİNDE AYNI PROPERTY'İ ALMAKTAN KAÇININ.



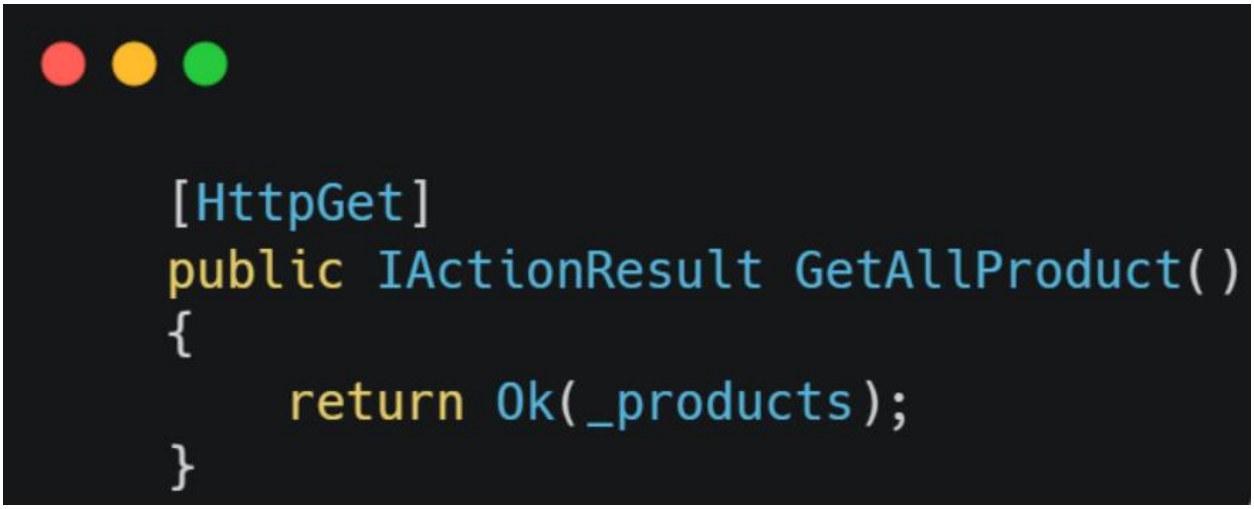
Program.cs Dosyasının mümkün olduğunda sade bırak!

Uygulama inşa edildikçe servisler artacak , katmanlar artacak ve bu iki kısımdaki kullanım da artmış olacak. Dolayısıyla Program.cs şişmeye başlar. ConfigureServices ve Configure yapıları extension metot içerisine taşımak gerekir. IServiceCollection içerisine extension metot yazıp çağırma işlemi gerçekleştirilebilir

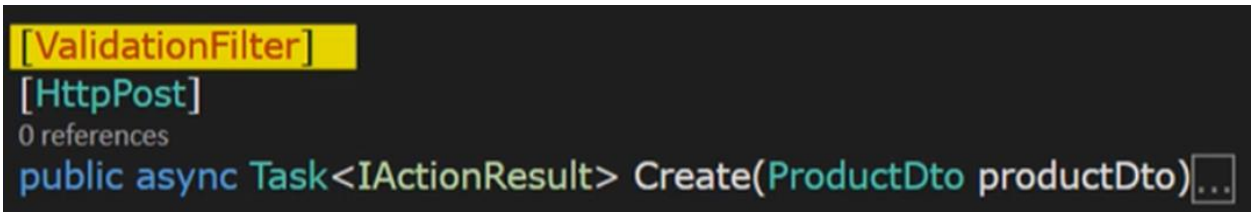
Uygulamanı Mümkün Olduğunca Küçük Parçalara Böl



Action Metotlarını temiz tutun Business Kodu bulundurmayın.



Hataları Global Olarak Ele Alın. Action metotlar içerisinde try catch blokları kullanmayın.



Entity Framework Core Code First Örneği

Bu proje, Entity Framework Core'un Code First yaklaşımını kullanarak basit bir öğrenci-sınıf ilişkisini simgeliyor.

Yüklenen Paketler

Entity Framework Core ve SQL Server paketlerini ekleyin: bash dotnet add package Microsoft.EntityFrameworkCore dotnet add package Microsoft.EntityFrameworkCore.SqlServer dotnet add package Microsoft.EntityFrameworkCore.Tools - Entity Framework Core ve SQL Server için gerekli paketler eklenir.

Kod Örnekleri

Model Sınıfları

Öğrenci ve Sınıf model sınıfları örnek:

Student.cs

```
public class Student
{
    [Key]
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Email { get; set; }
    public string Telephone { get; set; }
}
```

AppDbContext.cs

```
  public class AppDbContext : DbContext
  {
    public DbSet<Student> Students { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
```

```
    {
        optionsBuilder.UseSqlServer(@"Data Source=localhost;Initial
Catalog=Northwind;Integrated Security=True");
    }
  }
```

Program.cs

```
// See https://aka.ms/new-console-template for more information
using EntityFrameworkCore.Models;
using EntityFrameworkCoreExample.DataContext;



using (var context = new AppDbContext())
{
  //INSERT
  var student = new Student()
  {
    Email = "ruveydakardelcetin@gmail.com",
    FirstName = "Kardel",
    LastName = "Ruveyda",
    Telephone = "5305153061"
  };

  context.Students.Add(student);
  context.SaveChanges();
  Console.WriteLine("Veri tabanına başarılı bir şekilde eklendi!");
  Console.ReadLine();
```

```
//Tolist

var students = context.Students.ToList();

foreach (var item in students)
{
   Console.WriteLine(String.Join("Öğrenci adı : {0}", item.FirstName));
   Console.ReadLine();
}

////UPDATE
var studentUpdate = context.Students.Where(x => x.Id == 1).FirstOrDefault();

if (studentUpdate != null)
{
   studentUpdate.FirstName = "Aynur";
   studentUpdate.LastName = "Katırcıoğlu";

   context.Update(studentUpdate);
   context.SaveChanges();

   Console.WriteLine("Veri tabanına başarılı bir şekilde güncellendi");
   Console.ReadLine();
}
```

```
//Delete İşlemi

List<int> removeListIds = new List<int>();

removeListIds.Add(2);
removeListIds.Add(3);

bool isRemoved = false;
string message = "Bir hata oluştu";
try
{
   foreach (var item in removeListIds)
   {
      var studentDelete = context.Students.Where(x => x.Id == item).FirstOrDefault();
      if (studentDelete != null)
      {
         context.Students.Remove(studentDelete);
         context.SaveChanges();
         isRemoved = true;
         message = "Silme işlemi başarılı.";
      }
   }
}
catch (Exception)
{
```

```
        isRemoved = false;
    }



    if (isRemoved)
    {
        var studentNew = context.Students.Where(x => x.Id == 1).FirstOrDefault();

        if (studentNew != null)
        {
            studentNew.FirstName = "Test";
            studentNew.LastName = "Test2";
            studentNew.Email = "test3@gmail.com";
            studentNew.Telephone = "2124444444";

            context.Students.Update(studentNew);
            context.SaveChanges();

            message = "Hem silme hem güncelleme işlemi gerçekleşti.";
            isRemoved = true;
        }
    }
    Console.WriteLine(message);
    Console.ReadLine();
}
```