

# eBA Servis Entegrasyonlarına Giriş

21 Aralık 2020

Ayşegül YANIK

Yazılım Geliştirme Uzmanı

#### RESTFUL WEB SERVICE - SERVERSIDE & CLIENTSIDE

Çalışma 29 -Restful Servis Hakkında İlk Çalışmalar - 1

#### Çalışma Özeti

Servis entegrasyonları,

. eBa üzerinde geliştirilen projelerde Avicenna, ASMED, Octopus, HRWeb ve benzeri birçok farklı sağlayıcının sunduğu verileri alarak işlemek, kullanıcıya istenilen sonucun iletilmesinde temel bir ihtiyaçtır. Bunun sağlanmasında servis sağlayıcıları bizimle method tipi, gereken parametreler, url adres bilgisi, eğer varsa kimlik doğrulama bilgileri paylaşmaktadır. Bu bilgilerden yola çıkılarak veri akışının sağlanması, gelen verilerin karşılanması gidecek verilerin de uygun formatla hatasız olarak gönderimi için yürütülen çalışmadır.

#### **Rest Nedir?**

REST, servis yönelimli mimari üzerine oluşturulan yazılımlarda kullanılan bir transfer yöntemidir. İstemci ve sunucu arasında XML ve JSON verilerini taşıyarak uygulamanın haberleşmesini sağlar. REST mimarisini kullanan servislere ise RESTful servis denir.

# Web Servis Sağlayıcısı ve Tüketicisi Oluşturma Adımları : Rapor İçeriği

- ASP.Net Web Application (.Net Framework) projesi oluşturmak
- Web API şablonunun eklenmesi
- Aktarılacak verinin modelinin oluşturulması
- Varsayılan Controller kodunun özelleştirilmesi
- Model ile Controller 'ın Bağlanması
- Postman ve Swagger ile API Testi
- Client Tarafından Çağrı Yapılması

Açık Kaynak Paylaşılan Belgeler

> **Düzenleyen** Ayşegül YANIK

Yazılım Geliştirme Uzmanı

**Proje kodu** (2020) Bayındır Hastanesi -Yazılım Geliştirme - Proje 29 sayı: 029.0001/ayanik

## **RESTful Servis Oluşturmak**

Önce yeni bir ASP.Net Web Application (.Net Framework) projesi oluşturmak gerekir.

Create a new project

Recent project templates

Windows Forms App (NET Framework)

All platforms

All platforms

All platforms

All project types

All platforms

All project types

Apriject for creating a command-line application that can run on .NET Core on Windows, Linux maxOS windows Console

ASP.NET Core Web Application

Project templates for creating ASP NET Core web apps and web APIs for Windows, Linux maxOS windows Cloud Service Web

Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with favor paps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user larges (UIs).

Company in the browser on WebAssembly (wasm). These templat

Figure 1. Yeni ASP.Net Web Application (.Net Framework) Projesi

Yeni proje oluştururken, web API şablonu seçilirse yapılandırmalar yapılmış olarak proje açılacaktır.

Create a new ASP.NET Web Application Authentication No Authentication An empty project template for creating ASP.NET applications. This template does not have any content in it. A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag- and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access. Add folders & core references A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards. A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices. Advanced ✓ Configure for HTTPS Docker support (Requires Docker Desktop) Single Page Application A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5. CSS3, and JavaScript. Also create a project for unit tests Back Create

Figure 2. Web API Şablonu

## Controller Oluşturmak

Web servisler genellikle model - view - controller mimarisinde tasarlanır ancak bu olmazsa olmaz değildir. Bu tasarım örüntüsünde, kod ile yapılan işler fonksiyonlar işlevsel olan her şey Controller içinde yer alırken, verilerin yapılandırılarak saklandığı, veriler arasında bağlantıların kurulduğu yer Model alanıdır. View ise daha çok ön yüzün tasarımını ilgilendiren bir alan olarak programlama birbiri ile ilişkili üç faz halinde tamamlanır. Bu üçlü yapı sayesinde programlama kolaylaşır.

İkinci olarak yapılacak işlem, Controller tanımlamaktır. Burada projeyi oluşturur oluşturmaz gelen iki Controller mevcuttur. Boş proje yerine Web API seçmiş olduğumuzdan belli bir altyapı hazır sunulmuştur.

NESTWORD Serversion

Place Set View Gr. Project Bodd Deboy Set Analysis Bods Estension Window Help Search(Colific)

P RESTWORD Serversion

Nestword Set Set View Gr. Project Bodd Deboy Set Analysis Serversion

Figure 3. Default Controller

Varsayılan olarak yüklenen Controller içinde string olarak yüklenecek veriler tanımlanmıştır. Bu yapının yerine kendi verilerimizi eklemek mümkün. Bu noktada Model yapısından yararlanılır. Aktarılacak verinin yapısı birden çok değeri bir arada tutuyorsa, ona uygun bir yapı kurgulamak gerekecektir.

ccc Örnek olması açısından, ilk geliştirilen Servis API projesinde string tipindeki İsim Soyad bilgilerinin ve float decimal ve integer gibi farklı türdeki veri yapılarındaki değişkenlerin aktarımı ile ilgili bir senaryo tasarlandı.

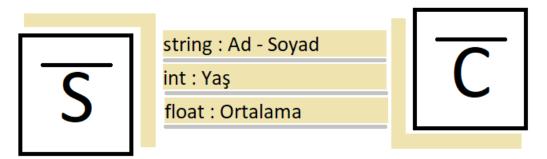


Figure 4. Server ile Client Arasında Tasınması Tasarlanan Veriler

Modeli oluşturmanın ilk adımı, modellerin arasına yeni bir sınıf eklemek.

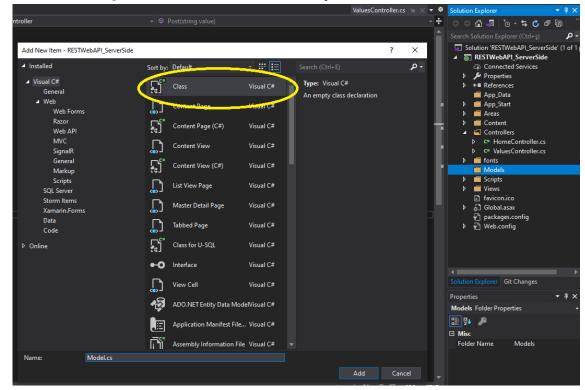


Figure 5. Server ile Client Arasında Taşınması Tasarlanan Veriler

Sınıf içeriği string tipli ad soyad, int tipli yaş ve float tipli ortalama bilgilerini tutsun istedim. Bu yüzden de sınıfın içeriğini aşağıdaki şekilde oluşturdum.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Web;
namespace RESTWebAPI_ServerSide.Models
   [DataContract]
   public class Model
      [DataMember(Name = "ad soyad")]
      public string AdSoyad { get; set; }
      [DataMember(Name = "yas")]
      public int Yas { get; set; }
      [DataMember(Name = "ortalama")]
      public float Ortalama { get; set; }
   }
}
```

## Model ile Controller arasındaki bağı oluşturmak

İki türde method kullanılarak çağrı yapılabilmektedir. Birincisi Get ikincisi POST. İlk örnek olarak Get tipinde bir metod ile yapılacak çağrının karşılığını servisime ekledim. Bu nedenle Post alanlarını Contorller alanından kaldırabilirim.

İlgili öğrencilerin bilgilerini aşağıdaki şekilde Controller içine ekledim.

```
public IEnumerable<Model> Get()
   var ogrenciListesi = new List<Model>();
   //Birinci rencinin Bilgileri
   var ogrenci_1 = new Model
   {
      AdSoyad = $"Ali Yank",
      Ortalama = 3.01f ,
      Yas = 19
   };
   ogrenciListesi.Add(ogrenci_1);
   //kinci rencinin Bilgileri
   var ogrenci_2 = new Model
      AdSoyad = $"Mertkan Yank",
      Ortalama = 3.29f,
      Yas = 28
   ogrenciListesi.Add(ogrenci_2);
             rencinin Bilgileri
   // nc
   var ogrenci_3 = new Model
      AdSoyad = $"Ayegl Yank",
      Ortalama = 3.12f,
      Yas = 27
   ogrenciListesi.Add(ogrenci_2);
   return ogrenciListesi;
}
```

## Servis Çalıştırma

Çalıştırıldığında, Üst Menüden API seçilir Get Values metoduna tıklanırsa aşağıdaki gibi bir şayfaya ulaşılır.

Figure 6. Değerlerin Tarayıcıdan Alınması

Görüldüğü üzere veri tipleri ve yapısıyla model sunulmuştur. Adres Çubuğunun sonundaki alana api/Values yazılırsa, (bu sonradan değiştirilebilir, bu kelimeler proje adı ve Conroller adından geldi) aşağıdaki şekilde tüm değerler tarayıcıda görülür.

Figure 7. Değerlerin Tarayıcıdan Alınması



#### Postman ve Swagger API Test

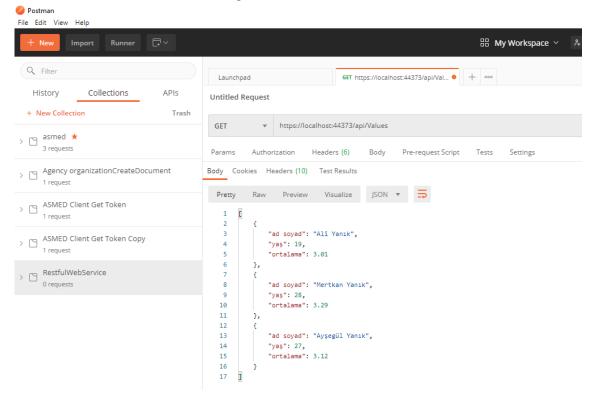
Postman uygulaması, swagger gibi servisin ne döndüreceğini ve ne şekilde döndüreceğini geliştiriciye sunan bir platformdur. Swagger açmak için, önce nuGet Packages içinden SwashBuckle paketi yüklenir. Daha sonra, adres çubuğuna localhost:XXXXX/swagger yazılır.

Figure 8. Değerlerin Tarayıcıdan Alınması



Postman ile test yapmak için ise, metod tipi seçilir URL yapıştırılır. Parametreler ve kimlik doğrulama bilgileri (varsa) ilgili alanlara girilir. Böylece aşağıdaki gibi bir sonuç alınır.

Figure 9. Postman Test



#### Client tarafından Bir Rest Servisin ConSume Edilmesi

Rest mimarisindeki bir servisin kullanıcı tarafından çağrılması sırasında postman ve swagger üzerinden yapılan işlem, .Net Framework yapısı kullanılarak gerçekleştirilir.

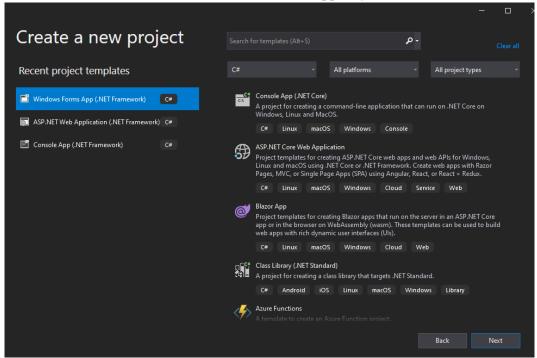


Figure 10. Yeni bir WindowsFormApp Projesi Oluşturma

Basit bir form ekranı tasarlanır, aşağıdaki kod kullanılarak çağrı yapılır.

```
using RestSharp;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp_ConsumeRESTAPI
  public partial class Form1 : Form
   { public Form1()
         InitializeComponent();
      private void button1_Click(object sender, EventArgs e)
         var client = new RestClient("https://localhost:44373/api/Values");
         client.Timeout = -1;
         var request = new RestRequest(Method.GET);
         IRestResponse response = client.Execute(request);
         textBox1.AppendText(response.Content);
      }
}
   }
```

Burada da yine bir paket ihtiyacı doğacaktır. Kodda hata alınan yerlerin üzerine gelerek tavsiyeleri takip edip gereken Nuget Paketlerinin eklenmesi tavsiye edilir. Burada RestSharp paketi gerekecektir. Uygulama çalıştırıldığında tasarım arayüzünde tasarlanan ve koda bağlanan alana response geldiği görülür.

Figure 11. .NET Framework üzerinden gelen Response

