bayındır

22 Aralık 2020

eBA Servis Entegrasyonlarına Giriş II

Ayşegül YANIK

Yazılım Geliştirme Uzmanı

RESTFUL WEB SERVICE - SERVERSIDE & CLIENTSIDE

Çalışma 29 -Restful Servis Hakkında İlk Çalışmalar - 2

Çalışma Özeti

Servis entegrasyonları 2,

. Bu çalışma, 029.0001 nolu çalışmanın devamıdır. Önceki çalışmada üç farklı öğrenci için ad soyad yaş ve ortalama bilgileri servis tarafından sunulmuş, client tarafından da karşılanarak kullanıcıya windows form arayüzünde sunulmuştu. Bu çalışmada ise, modele bir de resim eklenerek resmin binary formata çevrilmesi(serialization), string/binary olarak aktarılması ve karşı taraftan da uygun biçimde karşılanarak ve ve resim biçimine dönüştürülerek(deserialization) kullanıcıya windows form üzerinden sunulması gerçekleştirilmiştir. Bunun yanısıra raporda, get metodunun id ile (api/Value/id adres eklentisiyle) sıralı listeden seçili indexteki model verisinin içeriğinin çağrılmasına değinilmiştir. Bir sonraki (029.0003 nolu) çalışmada ise bu çağrıya parametrelerin ve header alanlarının eklenmesi hedeflenmektedir.

Giriş

Raporda aşağıdaki adımlar gerçeklenmiştir.

- Get metoduyla tüm listenin değil, listenin belirli indeksindeki model elemanının sunulması
- Modele Resim eklenmesi
- Eklenen Resmin serialize edilerek JSON formatında sunulması
- Client tarafında resmin JSON olarak alınıp deserialize edilerek png haliyle sunulması

Açık Kaynak
Paylaşılan Belgeler

Düzenleyen
Ayşegül YANIK

Yazılım Geliştirme Uzmanı

Proje kodu
(2020) Bayındır Hastanesi - Yazılım Geliştirme - Proje 29 sayı: 029.0002/ayanik

ID ile Get Metodu

Model içinde oluşturulan yapı (structure) sayesinde her yeni eklenen eleman bu yapıdan türeyen bir liste elemanı gibi eklenir. Her yeni eklenen elemana da bir id indeksi atanacaktır. Böylelikle URL alanından id ile sorgu yapılırsa çağrıya yalnızca o id 'de konuşlandırılmış olan model elemanı dönecektir.

Bunu tanımlarken aşağıdaki şekilde bir arguman alan get metodu yazılmalıdır.

```
public Model Get(int id)
{
   var ogrenciListesi = new List<Model>();

   //Birinci rencinin Bilgileri
   var ogrenci_1 = new Model
   {
      ID = 123,
      AdSoyad = "Ali Yank",
      Ortalama = 3.01f,
      Yas = 46,
   };
   ogrenciListesi.Add(ogrenci_1);

   //kinci rencinin Bilgileri
   var ogrenci_2 = new Model
   {
      ID = 234,
      AdSoyad = "Mertkan Yank",
      Ortalama = 3.29f,
      Yas = 28,
   };
   ogrenciListesi.Add(ogrenci_2);

   //   nc    rencinin Bilgileri
   var ogrenci_3 = new Model
   {
      ID = 345,
      AdSoyad = "Ayegl Yank",
      Ortalama = 3.12f,
      Yas = 27,
   };
   ogrenciListesi.Add(ogrenci_3);

   var ogrenci = ogrenciListesi[id];
   return ogrenci;
}
```

Bu metod çağrılırken de adres çubuğuna localhost:XXX/api/Values/id yazılarak çağrı yapılmalıdır. id yerine bu kod için 0,1 veya 2 yazılabilir. 0 Ali, 1 Mertkan, 2 Ayşegül öğrencilerinin bilgilerini getirecektir.

Modele Resim Eklenmesi

Models klasörüne eklenen class içinde yapılandırılan verimize bir de resim alanı eklendi.

```
//Model.cs icinde
[DataMember(Name = "Resim")]
public string Resim { get; set; }
```

Bu tür aktarımlarda resim png veya jpeg olarak aktarılmaz, çünkü verinin paketlenerek karşıya iletimi sırasında ne kadar gelişmiş ve yer tutan yapıdaki veri kullanılırsa o kadar güvenlik ihlali, zaman kaybı, veri kaybı vs ihtimalleri artacaktır. Bu yüzden veriler genellikle byte türüne çevrilerek iletilir; karşılandığı yerde yeniden uygun formata dönüştürülür.

Eklenen Resmin serialize edilerek JSON formatında sunulması

Projedeki modelde string olarak yapılandırılmasını uygun gördüm. Böylece byte ile string arasındaki dönüşümü de kod olarak raporuma eklemiş olmak istedim. Eğer model nesnesinin, resim (attribute) öğesini model içinde string yerine doğrudan byte olarak ekleseydim bu dönüşümü hiç kullanmadan da aktarımı sağlayabilirdim. Resmin formatlanması aşağıdaki kodla sağlandı.

```
//Resmin Formatlanmas >> Get metodu altnda
List<Image> image = new List<Image>();
image.Add(Image.FromFile("C:\\Images\\dandelion.png"));

byte[] bytes;
using (MemoryStream ms = new MemoryStream())
{
   BinaryFormatter formatter = new BinaryFormatter();
   formatter.Serialize(ms, image);
   bytes = ms.ToArray();
}
```

Yukarıdaki kod yardımıyla byte 'lara aktarılan resim model yapısında tanımlanan veriye de aşağıdaki gibi eklendi.

```
   var ogrenciListesi = new List<Model>();

   var ogrenci_1 = new Model
   {
      ID = 123,
      AdSoyad = "Ali Yanik",
      Ortalama = 3.01f,
      Yas = 46,
      Resim = BitConverter.ToString(bytes,0)
   };
   ogrenciListesi.Add(ogrenci_1);
```

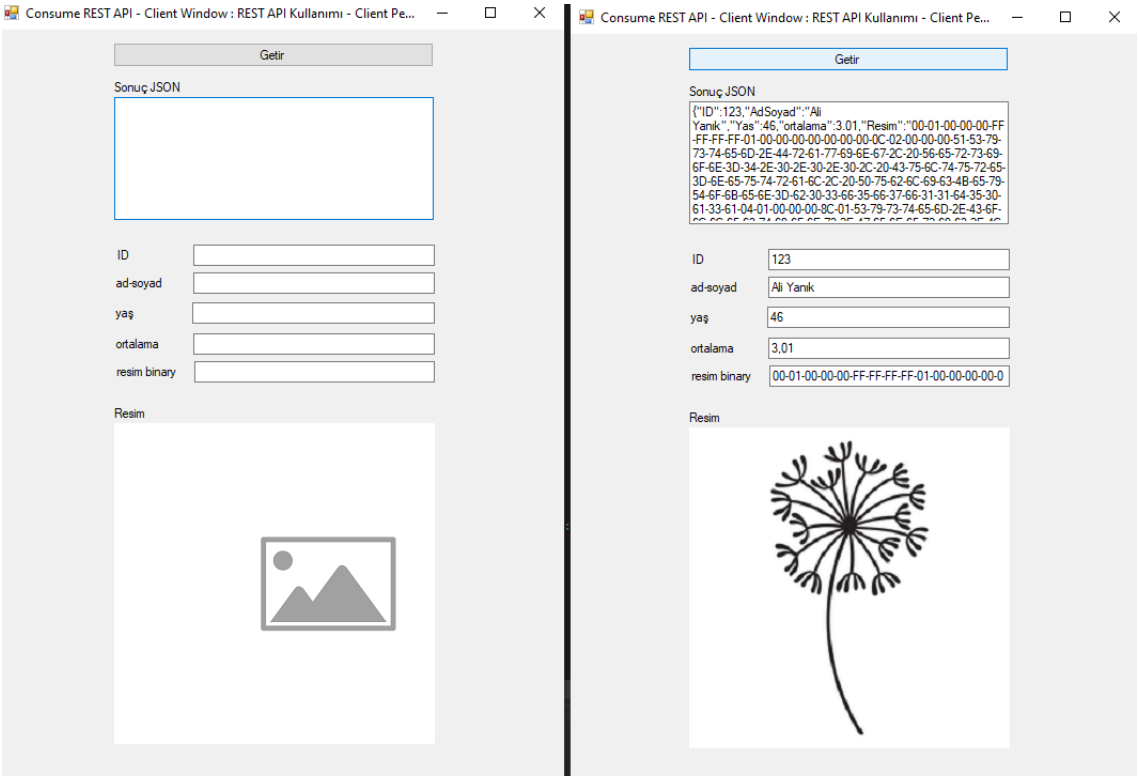
Verinin Deserialize Edilmesi

Sunulan veri sunucu tarafında aşağıdaki gibi görünür.

```
<Model xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.datacontract.org/2004/07/RESTWebAPI_ServerSide.Models">
<AdSoyad>Ali Yanik</AdSoyad>
<ID>123</ID>
<Resim>00-01-00-00-00-FF-FF-FF-FF-01-00-00-....-82-0B</Resim>
<Yas>46</Yas>
<ortalama>3.01</ortalama>
</Model>
```

Diğer yandan, Client tarafı için geliştirilen windows Form üzerinde ise aşağıdaki gibi görünecektir.

Figure 1. Client tarafından Form üzerinde Getir denilince gelen arayüz



Bunun sağlanması için, önce form (initialization) açılırken aşağıdaki gibi bir yer tutucu (placeholder) resim yüklendi.

```
public Form1()
{
   InitializeComponent();
   //image initialize placeholder - resim icin kapsam icinde bir varsayilan
   Image image = Image.FromFile("C:\\Images\\placeholder.png");
   pictureBox1.Image = image;
}
```

Getir butonuna tıklandığında ise aşağıdaki kod çalışacak ve önce gelen JSON türündeki yanıtı bir modele aktarıp (ad-soyad, yaş, ortalama, ID, binary haldeki resim) öğelerine ayıracak, sonra ilgili nesneye ait resmi binary formattan png formatına çevirerek resim alanına kaynak gösterecektir.

```
private void button1_Click(object sender, EventArgs e)
{
   var client = new RestClient("https://localhost:44373/api/Values/0");
   client.Timeout = -1;
   var request = new RestRequest(Method.GET);
   IRestResponse response = client.Execute(request);

   // Ornek Yanit >>
   //{"id":123,"adsoyad":"AliYanik","yas":46,"ortalama":3.01,"resim":"00.."}
   string result = response.Content;
   textBox1.Text = result;

   Model ogrenci = new Model();
   ogrenci = JsonConvert.DeserializeObject<Model>(result);

   textBox2.Text = ogrenci.AdSoyad;
   textBox3.Text = ogrenci.Ortalama.ToString();
   textBox4.Text = ogrenci.ID.ToString();
   textBox5.Text = ogrenci.Yas.ToString();
   string resimSTR = ogrenci.Resim;
   textBox6.Text = resimSTR;

   //String gelen resmi byte turune cevirmek
   String[] arr = resimSTR.Split('-');
   byte[] array = new byte[arr.Length];
   for (int i = 0; i < arr.Length; i++) array[i] = Convert.ToByte(arr[i],
       16);

   //resmi byte turunden image turune cevirmek DESERIALIZE
   using (MemoryStream ms = new MemoryStream(array))
   {
      BinaryFormatter formatter = new BinaryFormatter();
      List<Image> output_images = ((List<Image>)formatter.Deserialize(ms));
      Image image = output_images[0];
      pictureBox1.Image = image;
   }
}
```

İlgili kod ile sağlanan sonuç "Figure1.Client tarafından Form üzerinde Getir denilince gelen arayüz" isimli resimde incelenebilir.

Gelecek Çalışma

İlgili (Request) çağrıya parametre, başlık (header), kimlik doğrulama (authentication) gibi alanların eklenmesi ve daha kompleks çağrıların servis ile client arasıdna tasarlanarak kodlanması planlanmaktadır.