Rest ile SharePoint 2007-2016 Arası Veri Taşıma

İBRAHIM ERSOY

İÇİNDEKİLER

ÖNSÖZ

Verilerin bir ortamdan diğerine aktarılması firmalar için hep uğraştırıcı bir iş niteliğinde olmuştur.Bazen öyle bir hal alır ki veri aktarımı hayati önem taşır.

Sunucuların life support sürelerinin dolması,versiyon desteklerinin kısıtlanması/kaldırılması gibi konular IT ekiplerini versiyon güncelleştirmeye veya içeriklerini başka ortamlara/uygulamalara aktarmalarını zorunlu hale getirmiştir.

Bu e-kitabı yazarken amacım, bu işlemi lisanslı içerik aktarım araçlarını(migration tools) kullanmadan ücretsiz olarak nasıl gerçekleştirebileceğinizi göstermek olacaktır.

İçerik Aktarım Araçları'nın yapamadığı/uzun bir sürede yapabildiği iş akışlarının taşınması **bu yöntemin kapsamı dışındadır**.
İş akışlarının da taşınması istiyorsanız profesyonel bir İçerik Aktarım Aracı kullanmanızı tavsiye ederim.
Bunun dışında kalan tüm içeriklerinizi bu e-kitapta anlatacağım yöntem ile taşıyabilirsiniz.

Kitabı yazmamda bana ilham kaynağı ve destek olan tüm arkadaşlarıma teşekkürlerimi bir borç bilirim.

YAZAR HAKKINDA

İbrahim Ersoy,bilişim sektörüne adımını ilk olarak 2005 senesinde atmıştır.Çeşitli website tasarım ve programlama projelerinden sonra 2010 senesinde SharePoint ile tanışmıştır. SharePoint ürünü ile kurumsal intranetler,websiteleri üzerine danışmanlık hizmeti vererek çalışma hayatını sürdürmektedir.

23+ SharePoint Projesi'nde kurulumundan bakımına kadar görev alan yazar, şu anda "Araf Global" firmasında CTO pozisyonunda çalışma hayatına devam etmektedir.
Kendisi aynı zamanda "C# Corner MVP(4)" ödülü ile onurlandırılmıştır.
C# Corner ve UniftTurkiye teknik topluluklarında bilgisini paylaşmaktadır.

KAYNAK KODLAR

Kitaptaki örneklere ait kaynak kodlara;

https://github.com/IbrahimErsoy/REST_EKitap

Adresinden erişebilirsiniz.

1.BÖLÜM: NEDEN KOD İLE İÇERİK TAŞINIR?

Kod ile içerik taşıma konusuna girmeden evvel,neden içerik taşıyacağımızı konuşalım isterim.

Neden İçeriklerimizi taşımalıyız?

Microsoft'un her ürününde olduğu gibi ki bu firma bağımsız tüm sektördeki firmaların ürünleri için de geçerli,bir yaşam döngüsü mevcuttur.Zamanı gelince ürün niteliğini,niceliğini kaybeder.Ya firma üründen desteğini çeker ve zamanla ürün tamamen kullanımdan kalkar ya da ürün yeni bir isim/yeni bir yöntem ile kendi kapasitesini yenileyerek hayatına devam eder.

Hayatın kuralı da bu değil mi zaten?

Desteğin çekilmesi durumu destek süresi bitiminden itibaren oluşacak sorunların firmayı yasak olarak bağlamayacağı anlamına da gelmektedir.Bu durumda firmalar mümkün mertebede en son destek zamanını beklemeden gerekli adımları atmalı ve kendilerini güncellemelilerdir.Desteği kesilen ürünlerin muhakkak güncellenmesi gerekir.

İçerikleriniz firmanızın işleyişini devam ettirmeniz için elinizdeki bilgilerdir ve bu bilgiler ışığında şirketler faaliyetlerini yürütür ve gelecek için bu bilgiler ışığında stratejiler geliştirirler.

Eğer desteği kesilen/kesilecek ürünler kullanıyor iseniz,içeriklerinizi bir üst versiyona ve varsa gelecek planlarınızı versiyonları kademeli kademeli artırarak en üst sürümü kullanmaya odaklı planlamalar yapılması şarttır.

Konumuz SharePoint olunca,halen daha SharePoint 2007/2010 On-Premise yapılarını kullanan firmalar varsa ki yoktur diye ümit etmek istiyorum,bir an evvel upgrade/migration planlaması yapmanız gerekmektedir.Güncel,sürekli değişim halinde olan teknolojiyi yakından takip etmeniz,ne kadar değişime/gelişime açık olduğunuza bağlı olabilir ancak her yeni gelişme sizlerin işlerinizi daha rahat,sorunsuz/problemsiz icra edebilmeniz ve takip edebilmeniz için piyasaya çıkarılmaktadır.

Bu sebeple,imkanlar elverdiğinde içeriklerinizi bir ortamdan diğerine taşımanızın gerekli olduğu durumlar ortaya çıkacaktır.

SharePoint'te bir üst versiyona içerik taşıma işlemi bir çok yöntem ile yapılabilmektedir.

1-İçerik Taşıma Araçlarını kullanarak

2-Microsoft'un Aracını kullanarak yükseltme işlemi(versiyon güncellenir,içerikler olduğu şekilde kalır)

3-Kod ile içerik taşıma

Bu e-kitapta 3.yöntem olan kod ile taşıma anlatılacak

Kod ile İçerik Taşıma

Kod ile içerik taşınırken,kendinize sormanız gereken sorular vardır.Bu sorular aslında size neden kod ile içerik taşıma yöntemini seçmeniz gerektiği ile ilgili ipuçları da verecektir.

1-Her iki tarafta(kaynak ve hedef) da aynı yetkiye sahip aynı kullanıcı var mıdır?

Bu önemlidir çünkü bir ortamdan diğerine içerik aktardığınızda bu işlemi bir yönetici yetkisine sahip kullanıcı ile yapmalısınız.Bu kullanıcının her iki ortamda da aynı yönetici hakları(windows server admin grubuna

üyelik,sql server db izinleri-roller,sharepoint site koleksiyonu yöneticisi vb..) olması gerekir.

2-Hangi method ile içerik taşınacak?

Server Side REST mi WebService ASMX mi?

Tabi bunu seçerken versiyonun yeteneklerini de keşfetmek lazım.

3-İş Akışları?

İş Akışları birer "içerik" olarak değerlendirilmez.Dolayısıyla da İş Akışlarını kod ile taşıyamazsınız.

İş Akışları daha çok Collaboration ve Internal Process diye tabir ettiğimiz iş süreçleri ve birlikte çalışılabilirlik ortamları olarak belirtilir.İş Akışları taşımak istiyorsanız bir üst konu başlığında belirttiğimiz 1 ve 2 no'lu yöntemleri denemek zorundasınız.

4-Bir listeyi ne kadar zamanda taşırım?

Bu soru aslında listenin büyüklüğüne göre de değişebiliyor.Eğer versiyonları,yetkilendirmeleri,oluşturma/değiştirme/oluşturma tarihi/değiştirme tarihi gibi veriler de taşınacaksa,bir listenin taşınması sahip olduğu içerik sayısına ve sütun sayısına göre değişir.

Bu soruların cevabını verdikten sonra kafanızda nasıl bir araç geliştirmek istediğinize yönelik bir yapı oluşacaktır.

Kitabın kapsamı REST API ile içerik taşıma olduğundan,kullanacağımız method Sunucu Taraflı kod yazarak REST ile içerikleri taşımak olacaktır.

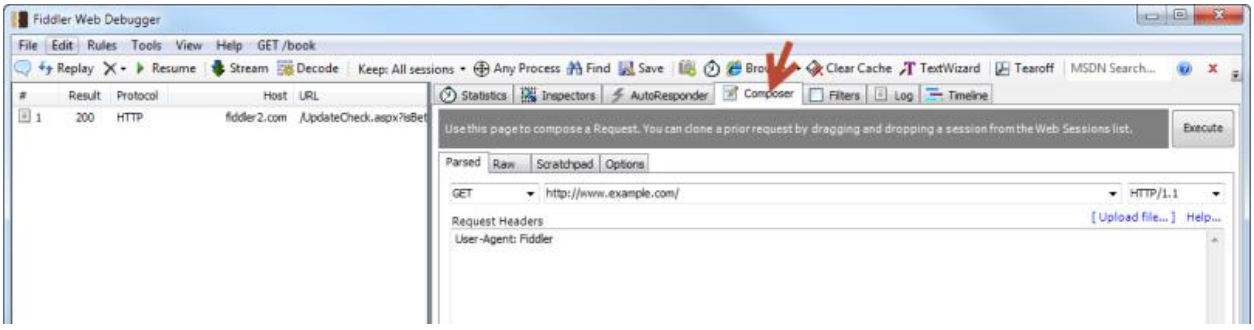
2.BÖLÜM: MERHABA,REST API!

SharePoint REST API' sinden bahsederken, REST(Representational State Transfer) 'in genel tarihine inmeden,siz değerli okuyucuları sıkmamak adına ne olduğuna dair kısa kısa bilgiler vereceğim.

REST, istemci ve sunucu arasında bağ kuran bir servis teknolojisidir.Yapmak istediğiniz işlemleri HTTP Methodlarına (GET,POST,PUT,DELETE) istekte bulunarak gerçekleştirirsiniz.

REST API bir Url üzerinden çalışır.siz bir rest çağrısında işlem yapmak istediğinizde bu url(web adresi) bilgisine istekte bulunursunuz.

ASMX bir webservise bağlanmak istediğinizde; http://testsunucusu.com/servis.asmx?wsdl şeklinde bir url kullanırken,
REST'te http://testsunucusu.com/servis şeklinde bir url kullanmaktayız.



(REST API çağrılarını Fiddler aracıyla izleyebilirsiniz)

SharePoint'te REST çağrılarından örnek vermemiz gerekirse;

http://test-sp/_api/contextinfo	Test-sp sunucusuna REST bağlanırken ilk olarak auth bilgisi almamız gerekmektedir.İlgili url'e POST çağrısı ile dönen guid bilgisi bizim auth anahtarımız,tokenımız olacak ve bu anahtar ile dilediğimiz işlemleri yapabileceğiz.
http://test-sp/_api/web/lists(guid'" + listId + "')	Test-sp sunucusundan guid değeri "listId" olan liste bilgisi çekilmekte.Burada tabi listId değerinin yerine bir list guid bilgisi gönderilmesi gerekmektedir.
http://test-sp/_api/web/lists/getByTitle('" + listname + "')	Test-sp sunucusundan liste başlığı "listname" olan liste bilgisi çekilmekte.listname yerine listenin başlığı girilmelidir.

Yukarıda verilen örnekler SharePoint REST Çağrılarından sadece bir kaçı.

Şimdi biraz kodlama üzerinde çalışalım:

Kodlamaya Geçiş

2015 yılında bir extranet projesinde kullandığım RestClient koduna aşağıdan erişebilirsiniz.Bu kodu REST Çağrıları yaparken kullanmaktayız:

```
public static void RestYap(string data, string digesturl, string requesturl, string listrequesturl,
bool ifmatch, bool merge, bool delete, bool put)
    {
        XmlNamespaceManager xmlnspm = new XmlNamespaceManager(new NameTable());
        xmlnspm.AddNamespace("atom", "http://www.w3.org/2005/Atom");
        xmlnspm.AddNamespace("d",
"http://schemas.microsoft.com/ado/2007/08/dataservices");
        xmlnspm.AddNamespace("m",
"http://schemas.microsoft.com/ado/2007/08/dataservices/metadata");

        HttpWebRequest contextinfoRequest =
(HttpWebRequest)HttpWebRequest.Create(digesturl);
```

```
        contextinfoRequest.Method = "POST";
        contextinfoRequest.Proxy = new WebProxy();
        contextinfoRequest.ContentType = "text/xml;charset=utf-8";
        contextinfoRequest.ContentLength = 0;
        contextinfoRequest.Credentials = CredentialCache.DefaultCredentials;

        HttpWebResponse contextinfoResponse =
(HttpWebResponse)contextinfoRequest.GetResponse();
        StreamReader contextinfoReader = new
StreamReader(contextinfoResponse.GetResponseStream(), System.Text.Encoding.UTF8);
        var formDigestXML = new XmlDocument();
        formDigestXML.LoadXml(contextinfoReader.ReadToEnd());
        var formDigestNode = formDigestXML.SelectSingleNode("//d:FormDigestValue",
xmlnspm);
        string formDigest = formDigestNode.InnerXml;

        HttpWebRequest listRequest =
(HttpWebRequest)HttpWebRequest.Create(listrequesturl);
        listRequest.Method = "GET";
        listRequest.Proxy = new WebProxy();
        listRequest.Accept = "application/atom+xml";
        listRequest.ContentType = "application/atom+xml;type=entry";
        listRequest.Credentials = CredentialCache.DefaultCredentials;
        HttpWebResponse listResponse = (HttpWebResponse)listRequest.GetResponse();
        StreamReader listReader = new StreamReader(listResponse.GetResponseStream());
        var listXml = new XmlDocument();
```

```
        listXml.LoadXml(listReader.ReadToEnd());


        var entityTypeNode =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:ListItemEntityTypeFullN
ame", xmlnspm);
        var listNameNode   =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:Title", xmlnspm);


        string entityTypeName = entityTypeNode.InnerXml;
        string listName = listNameNode.InnerXml;


        string itemPostBody = data;
        Encoding utf8NoBom = new UTF8Encoding(false);
        Byte[] itemPostData = utf8NoBom.GetBytes(itemPostBody);
        var datak = utf8NoBom.GetString(itemPostData);
        HttpWebRequest itemRequest = (HttpWebRequest)HttpWebRequest.Create(requesturl);
        itemRequest.Method = "POST";
        //itemRequest.Proxy = new WebProxy("127.0.0.1", 8888); İsteğinizin Fiddler'a düşmesini
sağlamak için local makinenizin 8888 portunu kullanmalısınız.
        //itemRequest.ContentLength = itemPostBody.Length;
        itemRequest.ContentLength = utf8NoBom.GetByteCount(datak);
        itemRequest.ContentType = "application/json;odata=verbose";
        itemRequest.MaximumResponseHeadersLength = -1;
        itemRequest.Proxy = new WebProxy();
        itemRequest.Accept = "application/json;odata=verbose";
        itemRequest.Credentials = CredentialCache.DefaultCredentials;
```

```
        itemRequest.Headers.Add("X-RequestDigest", formDigest);


        if (merge)
        {
           itemRequest.Headers.Add("X-HTTP-Method", "PATCH");
        }
        if (ifmatch)
        {
           itemRequest.Headers.Add("IF-MATCH", "*");
        }
        if (delete)
        {
           itemRequest.Headers.Add("X-HTTP-Method", "DELETE");
        }


        if (put)
        {
           itemRequest.Headers.Add("X-HTTP-Method", "PUT");
        }
        /// karşılaştırma file upload da if match ve delete true ise  yeni oluşturulacak  aksi
taktirde üzerine yazıcak
        if (ifmatch && delete)
        {
           itemRequest.ContentLength = 0;
        }
```

```
        else
        {
            Stream itemRequestStream = itemRequest.GetRequestStream();
            itemRequestStream.Write(itemPostData, 0, itemPostData.Length);
            itemRequestStream.Close();
        }
    }
```

Şimdi bu rest methodumuza ait birkaç örnek inceleyelim.Nasıl çalıştığı hakkında bilgi sahibi olacaksınız.

RESTClient'ımızın methodunun aldığı parametreleri inceleyelim:

```
RestHelper.RestYap("{'__metadata': {'type': 'SP.Data.TestListItem'},
                'Title': '" + item.Title + "'}",
                sp2016url + "_api/contextinfo",
                sp2016url + "_api/web/lists/GetByTitle('Test')/items",
                sp2016url + "_api/web/lists/GetByTitle('Test')",
                false, false, false, false);
```

1-) `"{'__metadata': {'type': 'SP.Data.TestListItem'},'Title': '" + item.Title + "'}"`

gönderdiğimiz ilk değerimiz POST edilecek isteğin gövdesi.Bir isteği POST ettiğinizde gövdesini de göndermeniz gerekmektedir."SP.Data.TestListItem" şeklinde ifade edilen yapı listemizin entitytypename'idir.ve type özelliği ile birlikte gönderilmelidir.Bu ifadenin ardından istediğiniz kadar sütun bilgisini virgule ayırarak gönderebilirsiniz.

Örneğin:

```
{'__metadata': {'type': 'SP.Data.TestListItem'},
 'Title': '" + item.Title + "',
 'ID':'" + item.ID + "',
 'Rol':'" + item["Rol"] + "',
 'User':'" + newuserid + "' }
```

şeklinde sütunlarınız varsa ekleyip gönderebilirsiniz.

2-) sp2016url + "_api/contextinfo"

bu değeri POST ettiğinizi ve dönen token bilgisi ile işlem yapmaya devam edeceğinizi bildiriyorsunuz.

Gelin bu token'ı daha önce oluşturduğumuz RESTClient'ımızda bulalım

```
HttpWebRequest contextinfoRequest = (HttpWebRequest)HttpWebRequest.Create(digesturl);
contextinfoRequest.Method = "POST";
contextinfoRequest.Proxy = new WebProxy();
contextinfoRequest.ContentType = "text/xml;charset=utf-8";
contextinfoRequest.ContentLength = 0;
contextinfoRequest.Credentials = CredentialCache.DefaultCredentials;

HttpWebResponse contextinfoResponse =
(HttpWebResponse)contextinfoRequest.GetResponse();
StreamReader contextinfoReader = new
StreamReader(contextinfoResponse.GetResponseStream(), System.Text.Encoding.UTF8);
var formDigestXML = new XmlDocument();
formDigestXML.LoadXml(contextinfoReader.ReadToEnd());
var formDigestNode = formDigestXML.SelectSingleNode("//d:FormDigestValue", xmlnspm);
string formDigest = formDigestNode.InnerXml;
```

Koddan da görüldüğü üzere methodumuzun 2.parametresinde gönderdiğimiz sp2016url + "_api/contextinfo" adresine POST işlemi gerçekleştiriyoruz.

Farkettiniz mi bilmiyorum ama POST çağrısında gövde göndermedik.Eğer ki POST Çağrısında gövde göndermeye ihtiyacınız yok ise,HttpWebRequest çağrınıza ContentLength = 0 ifadesini belirtmelisiniz.

Bu POST çağrınızın gövdesinin olmayacağı anlamına gelmektedir.

Gövdesi olan çağrılarda ise ContentLength gönderilen verinin byte cinsinden uzunluğu olmak zorundadır.Bu konuya ileride geri döneceğiz.

En son olarak XML node okuma işlemi gerçekleştirerek,d:FormDigestValue isimli değeri bir string değişkenine atıyoruz.

Bu değişkeni de RestClient'ımızın başlık bilgisine ek olarak ekliyoruz ki göndereceğimiz çağrı'nın token sahibi bir kullanıcı tarafından yapıldığını SharePoint anlayabilsin

```
itemRequest.Headers.Add("X-RequestDigest", formDigest);
```

şeklinde.

3-)
```
sp2016url + "_api/web/lists/GetByTitle('Test')/items",
sp2016url + "_api/web/lists/GetByTitle('Test')",
```

Bu iki parametreyi birlikte anlatma taraftarıyım.

İlk parametremiz "requesturl",ikincisi "listrequesturl"

Requesturl parametremiz direkt olarak list eleman işlemleri gerçekleştirmek için gerekli url bilgisini belirtirken,ListRequestUrl listemize ait özelliklere erişebilmek amacıyla sadece listemizin kendisinin bilgisini belirten url bilgisini ifade etmektedir.

Listrequesturl:

```
HttpWebRequest listRequest = (HttpWebRequest)HttpWebRequest.Create(listrequesturl);
listRequest.Method = "GET";
listRequest.Proxy = new WebProxy();
listRequest.Accept = "application/atom+xml";
listRequest.ContentType = "application/atom+xml;type=entry";
listRequest.Credentials = CredentialCache.DefaultCredentials;
HttpWebResponse listResponse = (HttpWebResponse)listRequest.GetResponse();
StreamReader listReader = new StreamReader(listResponse.GetResponseStream());
var listXml = new XmlDocument();
listXml.LoadXml(listReader.ReadToEnd());

var entityTypeNode =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:ListItemEntityTypeFullN
ame", xmlnspm);
var listNameNode   =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:Title", xmlnspm);
string entityTypeName = entityTypeNode.InnerXml;
string listName = listNameNode.InnerXml;
```

Listeye ait listeadı ve listentitytypename bilgilerini listrequesturl olarak gönderdiğimiz url ile almaktayız

Requesturl:

```
HttpWebRequest itemRequest = (HttpWebRequest)HttpWebRequest.Create(requesturl);
itemRequest.Method = "POST";
itemRequest.ContentLength = utf8NoBom.GetByteCount(datak);
itemRequest.ContentType = "application/json;odata=verbose";
itemRequest.MaximumResponseHeadersLength = -1;
itemRequest.Proxy = new WebProxy();
itemRequest.Accept = "application/json;odata=verbose";
itemRequest.Credentials = CredentialCache.DefaultCredentials;
itemRequest.Headers.Add("X-RequestDigest", formDigest);
```

RequestUrl ile yapacağımız 2.POST işlemine ait url bilgisini kullanmaktayız.

4-) false, false, false, false

Şimdi gelgelelim en önemli kısımlardan birine.Bu evet-hayır verisi içeren parametreler aslında bazı başlık bilgilerinin gönderilip gönderilmeyeceğini belirlememize yaramaktadır.

Nitekim,RESTClientımızı incelerseniz; bool ifmatch, bool merge, bool delete, bool put parametrelerini gönderdiğimizi göreceksiniz

Bu parametreler nasıl bir işlem yapmak istediğimiz ile ilgilidir.

POST mu PUT mu DELETE mi? Geleneksel CRUD olarak karşılıklarını verecek olsak,aşağıdaki gibi eşitlemeye benzeyecektir diye düşünüyorum.

POST : Add
PUT : Update
DELETE : Delete

Bu HTTP Methodlarını parametrelerimizde alacakları değere göre göndermekteyiz.

Örneğin:

ADD - POST	false, false, false, false
UPDATE – PUT	**true, true**, false, **true**
DELETE - DELETE	**true**, false, **true**, false

Şimdi de farklı sütunların nasıl gönderileceğini inceleyelim

3.BÖLÜM: HER SÜTUN TİPİ İÇİN REST ÇAĞRI GÖVDESİ

TEK/ÇOK SATIRLI METİN/SEÇENEK

Tek/Çok satır/Seçenek metinlerinde gövde kısmında yapılacak ekstra bir iş yükü yoktur.Sütun verisini olduğu gibi gönderebilirsiniz.

```
{'__metadata': {'type': 'SP.Data.TestListItem'},'Title': '" + item.Title + "'}
{'__metadata': {'type': 'SP.Data.TestListItem'},'Title': 'Merhaba Dünya'}
```

NÜMERİK DEĞERLER

Sayıları gönderirken de 2 farklı yöntem izlenir: Ondalıklı yöntem ve Tam Sayı Yöntemi

Tam Sayılar için;

```
{'__metadata': {'type': 'SP.Data.TestListItem'},'Order':" + item[“Order”] + "}
{'__metadata': {'type': 'SP.Data.TestListItem'},'Order':10}
```

Ondalıklı sayılar için;

```
{'__metadata': {'type': 'SP.Data.TestListItem'},'Order':'" + item[“Order”] + "'}
{'__metadata': {'type': 'SP.Data.TestListItem'},'Order':'10,256'}
```

Ondalıklı sayılarda kesme işaretinin kullanımına dikkat edin.Kesme işaretiyle ondalık sayı gönderme sebebimiz 10,3585 gibi bir ondalıklı sayıyı gönderdiğimizde ondalık virgülünü REST’in yeni bir sütun olarak algılamasını engellemek içindir.Tam sayılarda böyle bir zorunluluk yoktur.

RESİM

Bu sütun tipi için gövde değişmektedir.Diğer veri tiplerinde ListItem olarak özellik-değer belirtirken,resim tipinde bir içerik için farklı bir gövde göndermelisiniz.

{'__metadata': { 'type': 'SP.FieldUrlValue' }, 'Description': '" + picturevalue.Description + "','Url':'" + picturevalue.Url + "'}

{'__metadata': { 'type': 'SP.FieldUrlValue' },'Description': 'Test İmaj','Url':'www.x.com/logo.png'}

Sadece veritipini tanımladığımız alanda bu gövdeyi göndermeniz gerekmektedir.

Diğer sütunları da dahil ettiğimizde tam hali olarak;

{'__metadata': {'type': 'SP.Data.TestListItem'},'Title':'Merhaba',’Picture’: {'__metadata': { 'type': 'SP.FieldUrlValue' },'Description': 'Test İmaj','Url':'www.x.com/logo.png'} }

Şeklinde ana gövdeye eklenmelidir.

KÖPRÜ

Bu sütun tipi için de gövde değişmektedir.

{'__metadata': { 'type': 'SP.FieldUrlValue' }, 'Description': '" + navigationvalue.Description + "','Url':'" + navigationvalue.Url + "'}

{'__metadata': { 'type': 'SP.FieldUrlValue' }, 'Description': 'Açıklama','Url':'www.x.com'}

REST Çağrımıza eklemek istediğimizde ise;

{'__metadata': {'type': 'SP.Data.TestListItem'},'Title':'Merhaba',’Kopru’: {'__metadata': { 'type': 'SP.FieldUrlValue' }, 'Description': 'Açıklama','Url':'www.x.com'} }

Şeklinde bir gövde oluşturmamız gerekecektir.

ARAMA

Bu sütundaki tek arama değerini alabilmek için,string split yapmanız gerekecektir.

```
var parent = properties.ListItem["Parent"].ToString().Split(new string[] { ";#" },
StringSplitOptions.RemoveEmptyEntries)[0].ToString();
```

Parent isimli Arama sütunumuzdaki veriyi parent değişkeniyle aldıktan sonra REST üzerinden göndermek için sütun isminin sonuna “Id” ekini eklememiz gerekmektedir.

Tüm Arama,Kişi,Grup bilgilerini REST ile gönderirken “Id” ekini sütun adına eklemelisiniz

```
{'__metadata': {'type': 'SP.Data.TestListItem'},'Title':'Merhaba','ParentId':'" + parent + "' }
```

TARİH

Tarih değerlerinde çeşitli parametreler göndererek formatlama yaparsınız.

Örneğin;

```
'StartDate':'" + startdate.ToString("d") + "'
```

gönderdiğimizde tarih 10/14/2017 olacaktır (kısa tarih)

Veya

```
'StartDate':'" + startdate.ToString("D") + "'
```

Göndermek istediğimizde tarih Saturday, October 14, 2017 olur.

Bir ortamdan diğer ortama tarih bilgisini aktarmak için genelde round-trip desenini kullanmaktayız

```
'StartDate':'" + startdate.ToString("o") + "'
```

EVET/HAYIR

Evet/Hayır sütunu sadece 2 değer alabilmektedir.REST ile gönderirken;

'IsActive':" + aktifmi.ToString().ToLower() + "
'IsActive':EVET
'IsActive':HAYIR
'IsActive':evet
'IsActive':hayır

Şeklinde göndermemiz gerekir.

GRUP/KULLANICI

Grup/Kullanıcı için daha önceden de belirttiğimiz üzere sütun isminin sonuna "Id" ekleyerek ilgili Grubun/Kullanıcının ID değerini göndermeniz gerekmektedir.

{'__metadata': {'type': 'SP.Data.TestListItem'},'Title':'Merhaba','GonderenId':20 }

Gonderen isimli Grup/Kullanıcı Kabul eden sütuna 20 ID li Grup/Kullanıcıyı gönderiyoruz.

GRUPLAR/KULLANICILAR/ÇOKLU ARAMA

Bu sütunlar için REST gövdesinde dizi göndermeniz gerekmektedir.

{ '__metadata': { 'type': 'Collection(Edm.Int32)'},'results':[] }

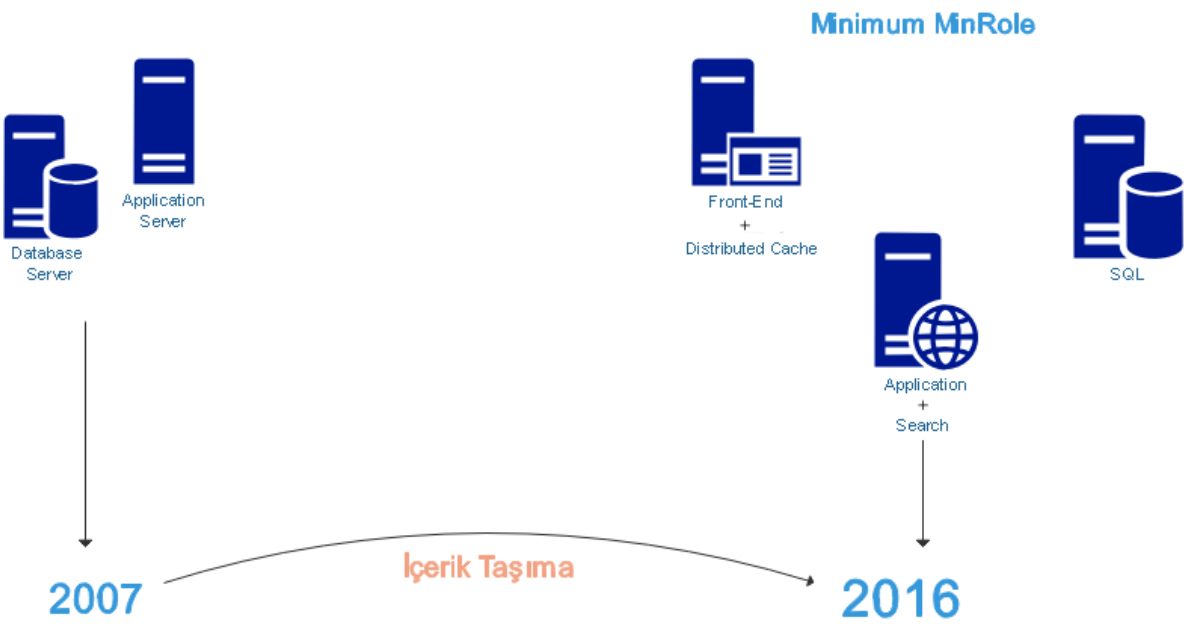
results dizisi içerisine Gruplar/Kullanıcılar'ın ID değerleri virgul ile ayırılarak eklenmelidir.

{ '__metadata': { 'type': 'Collection(Edm.Int32)'},'results':[31,13,24,54,79,65] }

Şimdi bir adım daha ileriye giderek,Kaynak-Hedef senaryomuza göre İçerik Taşıma işlemleri yapalım.

4.BÖLÜM: SHAREPOINT 2007'DEN SHAREPOINT 2016'YA VERİ AKTARIMI

Veri Taşıma ile ilgili yapımıza geçmeden evvel ortam hakkında senaryomuzu oluşturalım



REST ile SharePoint 2007-2016 Arası Veri Taşıma

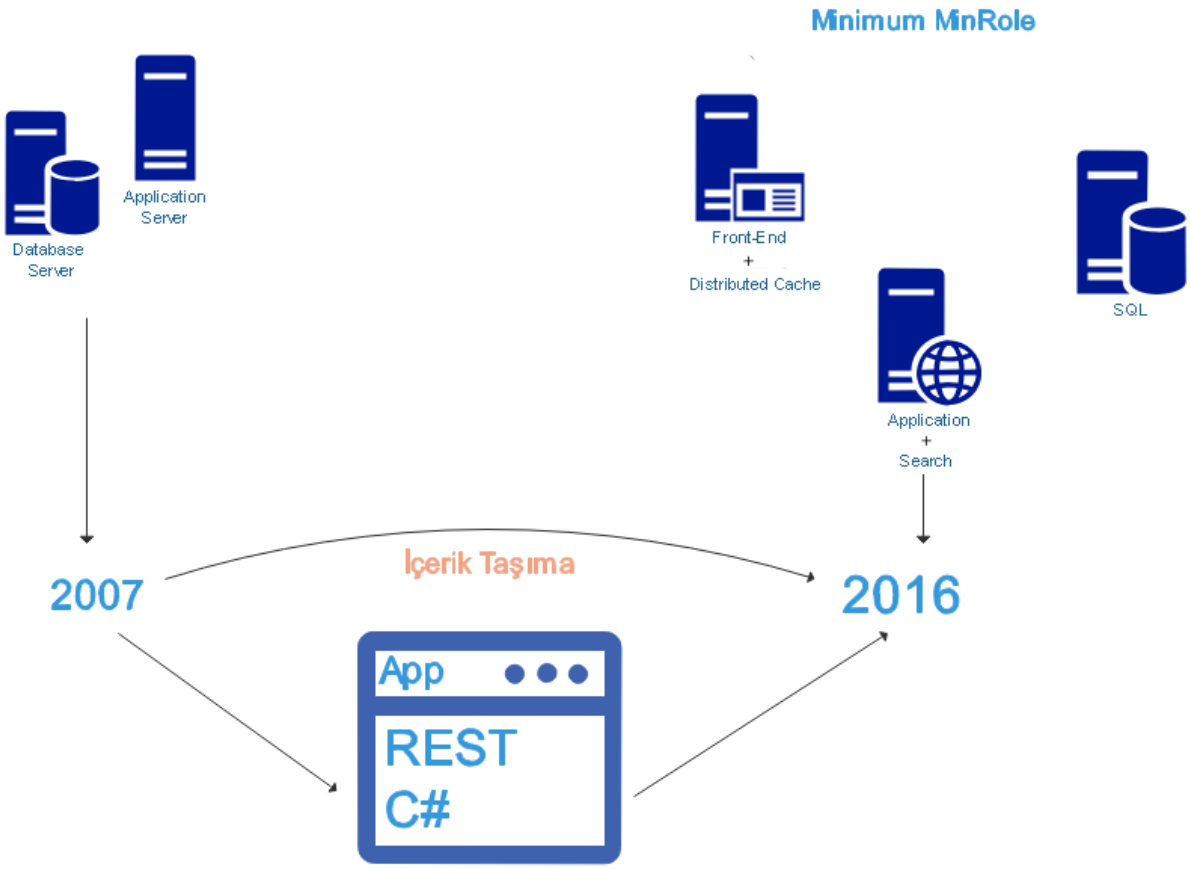
Senaryomuz şu şekilde işliyor olacak:

* X Firması,A Firmasından Danışmanlık Hizmeti alarak SharePoint 2007 Sunucularındaki içeriklerinin SharePoint 2016 sunucusuna direkt olarak taşınmasını istiyor.
* Firma SharePoint Server 2016 topolojisini yukarıda görüldüğü gibi kuruyor.
* Her iki ortamda da(tüm sunucularda) aynı kullanıcılar aynı yetkilere sahip(A firmasına özel deva kullanıcısı oluşturulmuş)
* Custom solution yok,iş akışları,formlar yok.Sadece listeler ve kütüphaneler'de saklanan içerik ve dosyalar var.
* Müşteriniz sizden içeriklerini SP 2016 ortamına aktarmanızı bekliyor.

Böyle bir yapıyı 2016'ya nasıl taşırsınız? 2007'den 2016'ya kadar oldukça köklü değişiklikler yapıldı.Versiyon farklılıkları mevcut.

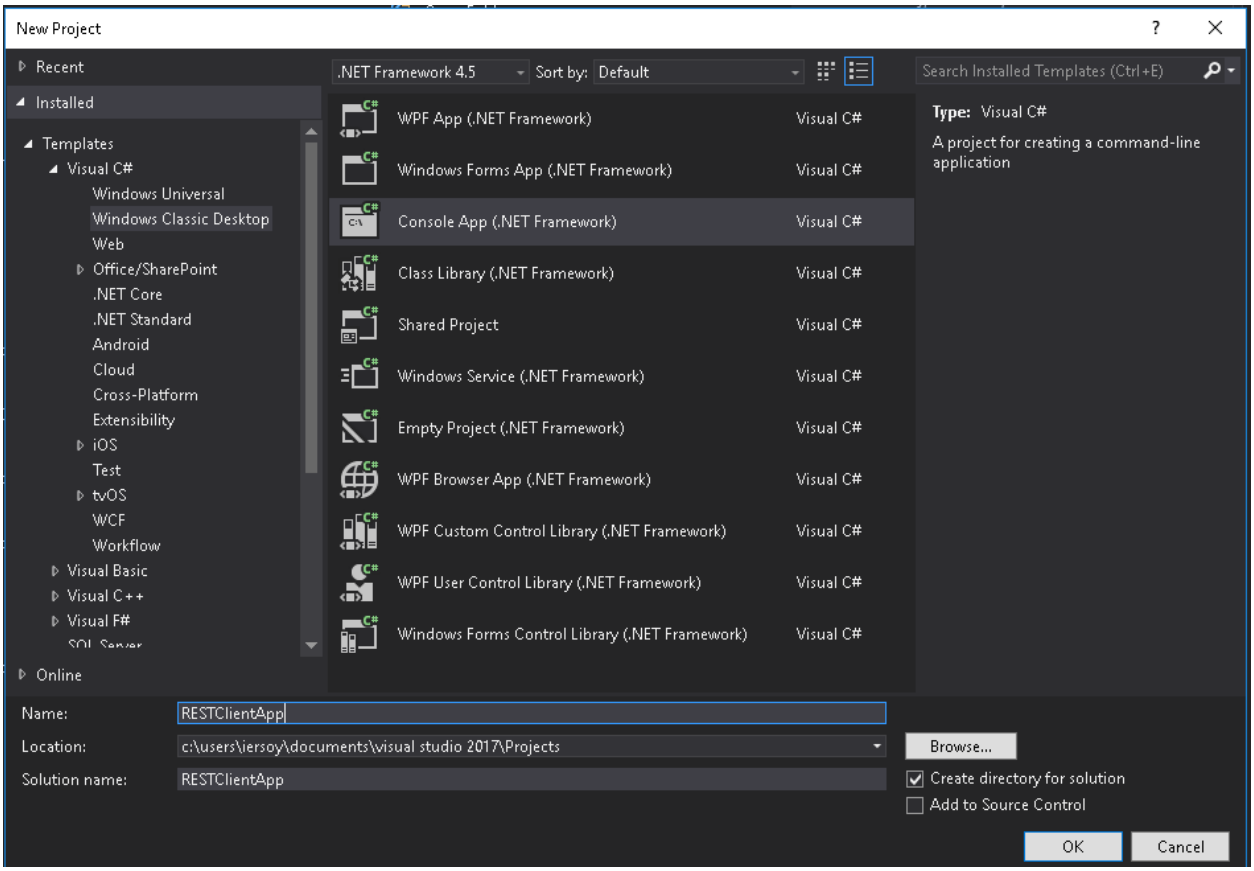
Bu durumda yapılacaklar basit: Ya bir Taşıma Aracı kullanıp içerikleri bir ortamdan diğerine aktaracaksınız ya da kendi custom taşıma betiği/uygulamanızı hazırlayacaksınız.

Custom uygulama olarak geliştirmeyi düşündüğünüzde en ideal tercih, SP 2007 ortamında çalışacak Sunucu Tabanlı C# REST/ASMX Konsol Uygulaması olacaktır.



O halde uygulamamızı oluşturmaya başlayalım.

Visual Studio 2013/2015/2017 (hangi versiyonu kullanıyorsanız) açarak yeni bir .NET Konsol Uygulaması oluşturun



Bu proje için “RESTClientApp” ismini verdim.

Proje oluştuktan sonra,2007 ortamından SharePoint referansını projeye eklemelisiniz.

SharePoint.dll genelde şu adreste bulunur: **C:\Program Files\Common Files\microsoft shared\Web Server Extensions\12\ISAPI\Microsoft.SharePoint.dll**

Ilgili .dll dosyasını projeye dahil ettikten sonra server side ile listelerden veri çekebiliriz.

ÖRNEK LİSTE : **DEPARTMANLAR**

String Title
String Sirket
Group Mudur

Pozisyonlar Listemizde LookUp(Arama değeri olacağından ötürü öncelikle Departmanlar Listesini taşımamız gerekmektedir.

Başlamadan evvel,bize yardımcı olacak bazı methodları barındıracağımız bir sınıfa ihtiyacımız var.İsmine Helper ve RestHelper diyerek 2 adet sınıf oluşturalım:

Helper.cs dosyamıza:

```
public static string Get_String(string fieldname, SPListItem item, SPWeb web, string sp2016url,
string restfieldId)
{
  var property = "";
  if (item[fieldname]!=null)
  {
    property = "'" + restfieldId + "':'" + item[fieldname].ToString().Replace("'", "").Replace("\\", "
\\\\ ") + "'";
  }
  else
  {
    property = "'" + restfieldId + "':null";
  }
  return property;
}
```

Get_String methodumuzda iki adet Replace kullandık.Bunu kullanma sebebimiz string manipulasyonu yapma isteğimizden.REST'in kafasını karıştıracak bazı özel karakterleri değiştirmeye yönelik kod yazdık.Ters Kesme(\) işaretini REST ile gönderdiğimizde eğer bir başka karakter ile birleşiyorsa,örneğin \n şeklinde REST ile ilgili karakteri gönderdiğimizde bize çağrı hatası fırlatacaktır.REST bazı özel karakterler ile birlikte çalışamamaktadır.Bunu çözmenin en kestirme yolu karakterin bir kopyasını daha oluşturarak göndermektir.

Şimdi de Helper.cs dosyamıza Get_Group methodunu dahil edelim:

```
public static string Get_Group(string fieldname,SPListItem item,SPWeb web,string
sp2016url,string restfieldId)
{
 var property = "";
 if (item[fieldname] != null)
 {
   SPFieldUser iksorumluField = (SPFieldUser)item.Fields.GetField(fieldname);
   SPFieldUserValue iksorumluValue =
(SPFieldUserValue)iksorumluField.GetFieldValue(item[fieldname].ToString());
```

```
  if (iksorumluValue.User == null)
  {
   try
   {
    SPGroup iksorumlugrp = web.SiteGroups[iksorumluValue.LookupValue];
    int newuserid = RestHelper.GetIdOfUser(sp2016url + "_api/web/sitegroups/GetByName('" +
iksorumlugrp.Name + "')?$select=id",sp2016url + "_api/contextinfo", iksorumlugrp.Name);
    if (newuserid != 0)
    {
     property = "'" + restfieldId + "':'" + newuserid + "'";
    }
    else
    {
     RestHelper.AddGroup("{ '__metadata':{ 'type': 'SP.Group' }, 'Title':'" + iksorumlugrp.Name +
"' }",sp2016url + "_api/contextinfo","http://aku-wb6/_api/web/sitegroups");
     newuserid = RestHelper.GetIdOfUser(sp2016url + "_api/web/sitegroups/GetByName('" +
iksorumlugrp.Name + "')?$select=id",sp2016url + "_api/contextinfo", iksorumlugrp.Name);
     property = "'" + restfieldId + "':'" + newuserid + "'";
    }
   }
   catch (Exception ex)
   {
    Console.WriteLine(ex.Message);
    property = "'" + restfieldId + "':null";
   }
  }
  else
  {
   try
   {
    SPUser user = web.EnsureUser(iksorumluValue.User.LoginName);
    int user_id = RestHelper.GetIdOfUser(sp2016url +
"_api/web/siteusers(@v)?@v=%27i%3A0%23.w%7C" + user.LoginName + "%27",sp2016url +
"_api/contextinfo",user.LoginName);
    if (user_id!=0)
    {
     property = "'" + restfieldId + "':'" + user_id + "'";
    }
    else
    {
     property = "'" + restfieldId + "':null";
    }
   }
  }
```

```
  catch (Exception)
  {
    property = "'" + restfieldId + "':null";
   }
  }
  }
  else
  {
   property = "'" + restfieldId + "':null";
  }
  return property;
 }
```

Biraz Get_Group kodumuzu inceleyelim.

Bu methodumuzda biz hem Grup hem de Kullanıcı bilgisini sorgulamaktayız.İlgili içeriğin Grup mu yoksa Kullanıcı mı olduğunu belirttikten sonra 2 koşula bölüyoruz.Eğer girilen very bir Kullanıcı ise, ID değerini 2016'dan almak zorunda olduğumuz için(REST ile User/Group/LookUp verilerini gönderirken ID değeri gönderme zorunluluğundan ötürü) öncelikle 2016'dan aynı kullanıcıya ait değeri çekiyoruz.

Ardından eğer ID değerimiz varsa yani kullanıcı 2016 ortamında da aynı bilgi ile yer alıyorsa,gövdemize parameter olarak gönderiyoruz.Eğer ilgili kullanıcı yok ise null(boş) değeri gövde ile birlikte gönderiyoruz.

Eğer içerik bir Grup ise,öncelikle Grup IDsini sorguluyoruz.Eğer Grup varsa,gövde ile birlikte grup ID'sini gönderiyoruz,lakin yoksa grubu oluşturuyor ve ardından tekrar ID sorgulama yapıp bu sefer yeni grup oluştuğundan IDsini alarak gövdeye parametreli bir şekilde gönderimini sağlıyoruz.

Şimdi de RestHelper.cs dosyamızın kodunu ekleyelim:

```
public static string GetIdOfList(string req, string digesturl)
{
 XmlNamespaceManager xmlnspm = new XmlNamespaceManager(new NameTable());
 xmlnspm.AddNamespace("atom", "http://www.w3.org/2005/Atom");
 xmlnspm.AddNamespace("d", "http://schemas.microsoft.com/ado/2007/08/dataservices");
xmlnspm.AddNamespace("m","http://schemas.microsoft.com/ado/2007/08/dataservices/meta
data"     );

HttpWebRequest contextinfoRequest = (HttpWebRequest)HttpWebRequest.Create(digesturl);
contextinfoRequest.Method = "POST";
```

```
contextinfoRequest.Proxy = new WebProxy();
contextinfoRequest.ContentType = "text/xml;charset=utf-8";
contextinfoRequest.ContentLength = 0;
contextinfoRequest.Credentials = CredentialCache.DefaultCredentials;

HttpWebResponse contextinfoResponse =
(HttpWebResponse)contextinfoRequest.GetResponse();
StreamReader contextinfoReader = new
StreamReader(contextinfoResponse.GetResponseStream(), System.Text.Encoding.UTF8);
var formDigestXML = new XmlDocument();
formDigestXML.LoadXml(contextinfoReader.ReadToEnd());
var formDigestNode = formDigestXML.SelectSingleNode("//d:FormDigestValue", xmlnspm);
string formDigest = formDigestNode.InnerXml;
formdigest = formDigest;

HttpWebRequest listRequest = (HttpWebRequest)HttpWebRequest.Create(req);
listRequest.Method = "GET";
listRequest.Proxy = new WebProxy();
listRequest.Accept = "application/atom+xml";
listRequest.ContentType = "application/atom+xml;type=entry";
listRequest.Credentials = CredentialCache.DefaultCredentials;
HttpWebResponse listResponse = (HttpWebResponse)listRequest.GetResponse();
StreamReader listReader = new StreamReader(listResponse.GetResponseStream());
var listXml = new XmlDocument();
listXml.LoadXml(listReader.ReadToEnd());

var listguid= listXml.SelectSingleNode("//atom:feed//atom:entry/atom:id", xmlnspm);
return listguid.InnerXml;
}


public static void RestYap(string data, string digesturl, string requesturl, string listrequesturl,
bool ifmatch, bool merge, bool delete, bool put)

    {

        XmlNamespaceManager xmlnspm = new XmlNamespaceManager(new NameTable());

        xmlnspm.AddNamespace("atom", "http://www.w3.org/2005/Atom");

        xmlnspm.AddNamespace("d",
"http://schemas.microsoft.com/ado/2007/08/dataservices");
```

```
        xmlnspm.AddNamespace("m",
"http://schemas.microsoft.com/ado/2007/08/dataservices/metadata");


        HttpWebRequest contextinfoRequest =
(HttpWebRequest)HttpWebRequest.Create(digesturl);
        contextinfoRequest.Method = "POST";
        contextinfoRequest.Proxy = new WebProxy();
        contextinfoRequest.ContentType = "text/xml;charset=utf-8";
        contextinfoRequest.ContentLength = 0;
        contextinfoRequest.Credentials = CredentialCache.DefaultCredentials;


        HttpWebResponse contextinfoResponse =
(HttpWebResponse)contextinfoRequest.GetResponse();
        StreamReader contextinfoReader = new
StreamReader(contextinfoResponse.GetResponseStream(), System.Text.Encoding.UTF8);
        var formDigestXML = new XmlDocument();
        formDigestXML.LoadXml(contextinfoReader.ReadToEnd());
        var formDigestNode = formDigestXML.SelectSingleNode("//d:FormDigestValue",
xmlnspm);
        string formDigest = formDigestNode.InnerXml;


        HttpWebRequest listRequest =
(HttpWebRequest)HttpWebRequest.Create(listrequesturl);
        listRequest.Method = "GET";
        listRequest.Proxy = new WebProxy();
        listRequest.Accept = "application/atom+xml";
```

```
        listRequest.ContentType = "application/atom+xml;type=entry";
        listRequest.Credentials = CredentialCache.DefaultCredentials;
        HttpWebResponse listResponse = (HttpWebResponse)listRequest.GetResponse();
        StreamReader listReader = new StreamReader(listResponse.GetResponseStream());
        var listXml = new XmlDocument();
        listXml.LoadXml(listReader.ReadToEnd());

        var entityTypeNode =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:ListItemEntityTypeFullN
ame", xmlnspm);
        var listNameNode   =
listXml.SelectSingleNode("//atom:entry/atom:content/m:properties/d:Title", xmlnspm);

        string entityTypeName = entityTypeNode.InnerXml;
        string listName = listNameNode.InnerXml;

        string itemPostBody = data;
        Encoding utf8NoBom = new UTF8Encoding(false);
        Byte[] itemPostData = utf8NoBom.GetBytes(itemPostBody);
        var datak = utf8NoBom.GetString(itemPostData);
        HttpWebRequest itemRequest = (HttpWebRequest)HttpWebRequest.Create(requesturl);
        itemRequest.Method = "POST";
        //itemRequest.Proxy = new WebProxy("127.0.0.1", 8888); İsteğinizin Fiddler'a düşmesini
sağlamak için local makinenizin 8888 portunu kullanmalısınız.
        //itemRequest.ContentLength = itemPostBody.Length;
        itemRequest.ContentLength = utf8NoBom.GetByteCount(datak);
```

```
itemRequest.ContentType = "application/json;odata=verbose";
itemRequest.MaximumResponseHeadersLength = -1;
itemRequest.Proxy = new WebProxy();
itemRequest.Accept = "application/json;odata=verbose";
itemRequest.Credentials = CredentialCache.DefaultCredentials;
itemRequest.Headers.Add("X-RequestDigest", formDigest);

if (merge)
{
   itemRequest.Headers.Add("X-HTTP-Method", "PATCH");
}
if (ifmatch)
{
   itemRequest.Headers.Add("IF-MATCH", "*");
}
if (delete)
{
   itemRequest.Headers.Add("X-HTTP-Method", "DELETE");
}

if (put)
{
   itemRequest.Headers.Add("X-HTTP-Method", "PUT");
}
```

```
        /// karşılaştırma file upload da if match ve delete true ise  yeni oluşturulacak  aksi taktirde üzerine yazıcak
        if (ifmatch && delete)
        {
            itemRequest.ContentLength = 0;
        }
        else
        {
            Stream itemRequestStream = itemRequest.GetRequestStream();
            itemRequestStream.Write(itemPostData, 0, itemPostData.Length);
            itemRequestStream.Close();
        }
    }
```

RestHelper ve Helper sınıflarımıza ait kodları ekledik ve ne işe yaradıklarını,hangi görevleri yerine getirdiklerini açıkladık.Şimdi Program.cs yani ana kod kısmına geçerek REST işlemlerimizi yapabiliriz.

Program.cs dosyamızda Standart SPSite-SPWeb açılımı yapalım. test-sp sunucusunda bulunan ik adlı subsite'a erişelim:

```
using (SPSite site = new SPSite("http://testsp2007"))
{
  using (SPWeb web = site.OpenWeb("/ik"))
  {


  }
}
```

1-Listemize erişelim:

SPList list = web.GetList(web.Url + "/Lists/Departmanlar");

2-ilk 5000 kaydı alacak sorgumuzu oluşturalım.

```
SPQuery query = new SPQuery();
query.RowLimit = 5000;
```

3-sp2016urlimizi bir değişkene atayalım:

```
string sp2016url = "http://testsp2016/ik";
SPUser kul = null;
```

4-2007'de bulunan listenin bir benzeri 2016 ortamında da yer almalı.Dolayısıyla listeninurl bilgisine erişebilmek için aşağıdaki gibi bir filtreleme sorgusu yazalım:

```
string listurl = "_api/web/Lists?$expand=ListItemAllFields&$filter=Title eq 'Departmanlar'";
string encodedlisturl = SPEncode.UrlEncode(listurl);
```

5-SP2007 ortamındaki liste guid'i ile 2016 ortamındaki liste guid'i aynı olmayabileceğinden bir rest çağrısı ile ilgili listenin ID'sini çekecek kodu ekliyoruz.

```
string listguid = RestHelper.GetIdOfList(sp2016url +
SPEncode.UrlDecodeAsUrl(encodedlisturl),sp2016url + "_api/contextinfo");
```

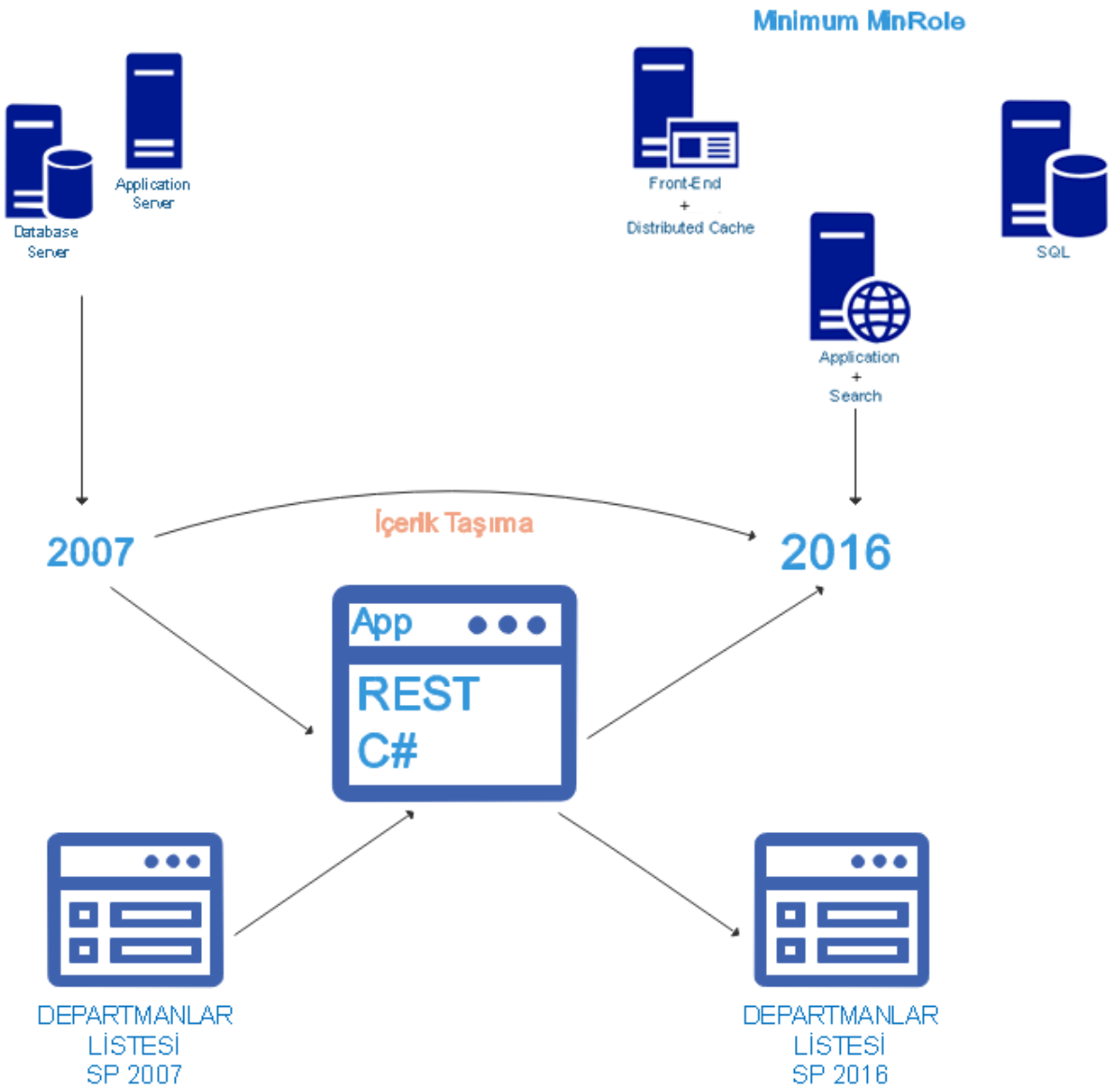
6-Verilerimizi çekmeye başlayalım

```
SPListItemCollection items = list.GetItems(query);
foreach (SPListItem item in items)
{
 try
 {
  if (item != null)
  {
   var sirket = Helper.Get_String("Sirket", item, web, sp2016url, "Sirket");
   var title = Helper.Get_String("Title", item, web, sp2016url, "Title");
   var mudur = Helper.Get_Group("Mudur", item, web, sp2016url, "MudurId");
   RestHelper.RestYap("{'__metadata': {'type': 'SP.Data.DepartmanlarListItem'}," + sirket + "," +
title + "," + mudur + " }",sp2016url + "_api/contextinfo",listguid + "/items",listguid,false, false,
false, false);
  }
 }
 catch (Exception ex)
```

```
 {
   Console.WriteLine(ex.StackTrace);
 }
}
```

7-Group/LookUp/User gibi verilerde Id gönderme zorunluluğumuz bulunmaktadır.Bu yüzden 2016 ortamından ilgili ID değerlerini öncelikle REST Çağrısı ile çekerek ardından parameter ile tekrar bir çağrıda bulunarak verilerimizi 2016 ortamına taşırız.

Aslında bu sistem OneWay (tek yönlü) bir entegrasyon değil,Mecburi TwoWay(Çift Yönlü) bir çalışma olmuştur.



Yukarıdaki mimari tasarımda da görüldüğü üzere dikkat edilmesi gereken husus,her iki ortamında veri bütünlüğünün olmasının zorunluluğu.

Kaynak ortamında(2007) olan bir liste tüm sütunlarıyla birlikte Hedef ortamda da (2016) olmak mecburiyetindedir.

Listelerin kendisini taşımak için 2007 şemasına bakıp field bilgilerini 2016 List Instance+Template olarak eklememiz gerektiğidir.2007 ortamında Listenin şema bilgisini alıp Visual Studio'da List Şablonunu oluşturup custom bir feature'a dahil edip 2016 ortamına Deployment işlemi yaptıktan sonra feature'ı enable ettiğimizde listeleri taşınmış bir şekilde

görebiliriz.Listelerin içerikleri eklenmeden taşınmasını Visual Studio ile SharePoint paketi hazırlayarak gerçekleştirebiliriz.

İçerikler için ise REST kullanarak istediğimiz verileri taşımamız mümkündür.

ID ATAMA

Liste Elemanlarında "ID Preservation" yani ID değerlerini de REST ile olduğu gibi taşımanız mümkündür.Bu konuda herhangi bir engel bulunmamaktadır.

LÂKİN;

ID'leri siz atadığınız zaman SharePoint'in kafası karışır ve manuel olarak listeye girip yeni eleman eklemek istediğinizde ekleyemezsiniz.SharePoint REST API de ID Manipulation Bug'ı bulunmaktadır.

Bu bug'ı keşfettiğimde aklıma gelen tek çözüm Event Receiver yazarak otomatik olarak SQL Database'inde ilgili tablolarındaki sütunun değerini update etmek oldu.

Bu konuyla ilgili çözümümüzü Microsoft'a da ilettik ve herkesin haberdar olması için de hem Türkçe hem de İngilizce olarak makalesini yazdık.Eğer ilginizi çektiyse,aşağıda linkini paylaşıyorum.

Makalenin Türkçesi için tıklayın
Makalenin İngilizcesi için tıklayın

SONSÖZ

SharePoint REST API gerçekten çok güçlü ve hızlı bir çözüm olarak karşımıza çıkmaktadır.SharePoint REST API piyasadaki kullanımına baktığınızda büyük bir yüzdesini içeriklerin liste/kütüphanelerden çekilerek gösterilmesi(GET Çağrıları) olduğunu göreceksinizdir.

Klasik geliştirme diye tabir ettiğimiz sunucu taraflı kodlamanın Intranet-Extranet-Internet portallarında yarattığı performans düşüklüğü sorununa çözüm getirecek olan, Client Side(İstemci Taraflı) betiklerin kullanılarak REST ile çok hızlı bir şekilde gösterilmesidir.

Sunucu taraflı çözümler Intranet portallarındaki kullanımını İstemci taraflı çözümlere çoktan bıraktı.Ancak,teknik bakım görevlerini ve istemci tarafında yapılamayacak birçok işlemi biz yine PowerShell komutlarıyla veya yazacağımız custom konsol uygulamalarıyla sunucu taraflı kodları kullanarak gerçekleştiriyor olacağız.

Sunucu taraflı kodlama OnPremise sistemler varoldukça olmaya devam edecektir.Ancak kullanımı son yıllarda istemci taraflı betik dillerinin güçlenmesi ve Node.js gibi sunucu tabanlı Javascript mimarisinin gelişmesi ve yaygın kullanılmasıyla çehresini değiştirmiş ve artık sadece işin arka plan mimarisinde,yeni teknolojilerin(Node.js,JS Kütüphaneleri) yeteneklerini aşan senaryolarda kullanılmaya devam edecektir.

Yazdığım e-kitabın konusu olan “REST ile farklı ortamlardan İçerik Taşıma” da sunucu taraflı kodlamanın önemini ve yeni teknolojiler ile olan bağını belirtmesi açısından önemli olduğunu düşünmekteyim.

Bu e-kitap size alternatifsiz kaldığınızda ışık olması ümidiyle yazdım.Umarım faydalı olabilmişimdir.

Mutlu ve Bol SharePoint’li Günler!

#YasasinSharePoint!