T.C.
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİNGÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

ETKİLEŞİMLİ WEB UYGULAMALARI-4

ANKARA,2008

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğrenme materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere mesleki ve teknik eğitim okul ve kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlık'ta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0093**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Web Programcılığı**
MODÜLÜN ADI	**Etkileşimli Web Uygulamaları-4**
MODÜLÜN TANIMI	XML uygulamaları yapmayı, web servisleri kullanmayı anlatan öğrenim materyalidir.
SÜRE	40/32
ÖN KOŞUL	Etkileşimli Web Uygulamaları 3 modülünü bitirmiş olmak
YETERLİK	Web servislerini kullanmak
MODÜLÜN AMACI	**Genel Amaç:** Web servislerini kullanabileceksiniz. **Amaçlar:** **1**.XML uygulaması yapabileceksiniz. **2**.Web servislerini kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuarı, internet
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz.

GİRİŞ

Sevgili Öğrenci,

Günümüzde iletişim yaşamın vazgeçilmez bir parçasıdır. Herkesin ortak bir dili kullanması için çeşitli denemeler yapılmıştır. Bu çabalar bir sonuca ulaşmamıştır ama zamanla İngilizce yaygınlaşarak insanların ortak anlaşma dili durumuna ulaşmıştır.

Bilişim dünyasında da programlar arasındaki iletişimin gerçekleşmesi, farklı sistemler arasında veri alışverişinin sorunsuz olması için ortak yapılar, standartlar oluşturma çabaları devam etmektedir. Bu çabalardan birisi XML dilidir.

Bu modülde verilerin platformdan bağımsız olarak dolaşımını sağlamak amacıyla üretilmiş XML dili ile web servislerini öğreneceksiniz.

ÖĞRENME FAALİYETİ–1

AMAÇ

XML uygulaması yapabileceksiniz.

ARAŞTIRMA

1. XML dilinin ne amaçla kullanıldığını araştırınız.

1. XML

1.1. XML Dili

XML, farklı bilgisayar sistemleri arasında veri iletmek için tasarlanmış yazılım ve donanımdan bağımsız bir dildir. XML **(Extensible Markup Language)** kavramının Türkçesi **Genişletilebilir İşaretleme Dili**'dir.

Bu dil bilgisayar dünyasında bazı ihtiyaçların giderilmesi amacıyla geliştirilmiştir. Veri aktarmak HTML sayfalarıyla yapılabiliyordu. Örneğin borsa verilerini dağıtan bir sayfadan bilgi almak mümkündü. İnternette her türlü veriye ulaşılabiliyordu fakat verileri tanımlamak, analiz etmek, yeniden biçimlemek mümkün değildi. İşte bu ihtiyaçları gidermek için istenildiği kadar genişletilebilen ve tekrar biçimlendirilebilen bir yapıya sahip XML geliştirildi.

XML uyumsuz **(incompatible)** sistemler arasında veri alışverişi için kullanılabilir. Bilgisayar programları uyumsuz formatlarda veri tutarlar. XML'nin kullanım alanları Internet uygulamaları ile sınırlı olmayıp geliştirilme amacı, birbiriyle bilgi alışverişi yapması istenen uygulamalar için ortak bir dil sunmaktır. Verinin XML'ye dönüştürülmesi, veriyi farklı türdeki uygulamalar (farklı işletim sistemleri üzerinde çalışan, farklı firmalar tarafından yazılmış, farklı dillerle yazılmış vs. uygulamalar) tarafından okunabilen veri haline getirerek veri alışverişinde yaşanan sıkıntıyı büyük ölçüde azaltır. XML belgeleri hangi uygulama tarafından kullanılacaksa biçimlendirilerek o uygulamaya uyumlu hale getirilir.

XML, verileri metin tabanlı tanımlar. XML belgeleri, verilerin etiketlenerek bir düzen içinde tutulduğu metin dosyalarıdır. XML ile düz metin dosyaları veri paylaşmak için kullanılabilir. XML verileri düz metin formatında saklandığından XML yazılım ve donanımdan bağımsız şekilde veri paylaşımını mümkün kılar. Bu durum farklı uygulamaların çalışabileceği verilerin oluşturulmasını kolaylaştırır.

XML belgeleri, verileri tutmak amacıyla kullanıldığından verilerin birbirleriyle olan ilişkileri de belirtilmelidir. Etiketleme işlemi verilerin birbiriyle ilişkilerini belirlemek için kullanılır. Örneğin, elimizde aşağıdaki veriler olsun.

Burak
Eren
Polis
Yunus
Hızır
Savcı

Bu verilere bir kimlik kazandırmak ve aralarındaki ilişkiyi belirlemek için etiketler kullanalım.

```
<personel>
      <eleman>
         <ad>Burak</ad>
         <soyad>Eren</soyad>
         <meslek>Polis</meslek>
      </eleman>
      <eleman>
         <ad>Yunus</ad>
         <soyad>Hızır</soyad>
         <meslek>Savcı</meslek>
      </eleman>
</personel>
```

Yukarıdaki örnekte "Burak, Eren, Polis, Yunus, Hızır, Savcı" bilgileri arasındaki ilişki belli değildir. "Eren" kelimesi ayrı bir ismi mi yoksa "Burak" isimli elemanın soyadını mı ifade etmektedir? "Polis" rastgele konmuş bir kelime midir yoksa Burak"ın mesleği midir? İşte bilgiler arasındaki doğru ilişkilendirmeyi kurmak için etiketlendirmeler kullanılmıştır. Dikkat edilirse HTML'ye benzer bir etiketlendirme mantığı kullanılmıştır.

HTML'den farklı olarak XML'de veri ile verinin sunumu (verinin istenilen biçimde gösterilmesi) birbirinden ayrıdır. Bu durum XML'nin yararlarından biridir. Böylelikle farklı stil sayfaları kullanarak XML belgesinden birçok biçimlendirilmiş sayfa üretebilir.

XML, HTML'nin türetildiği SGML **(Standart Generalized Markup Language)** dilinden türetilmiştir. HTML'de bütün tanımlamalar, adlar bellidir ve bunların dışına çıkılmaması gerekir. XML'de asıl önemli olan veridir. XML belgelerinde biçimlenmemiş veriler bulunur ve etiket adlarını belirlerken, belli temel kurallara bağlı kalmak koşuluyla herhangi bir tanımlayıcı isim kullanılabilir. XML, HTML gibi ücretsiz bir dildir.

Örnekte <personel>, <eleman>, <ad> gibi elemanlar kullanılmıştır. Bir başkası bu bilgileri tutmak için farklı eleman isimleri kullanabilirdi. Örneğin, <çalışanlar>, <çalışan>, <çalışan_ismi>. XML'nin yaygınlaşmasını sağlayan bu esnekliğin dezavantajı da vardır.

Farklı uygulamalar arasında veri alışverişinde farklı elemanların kullanılmış olması sıkıntılara neden olur. Veriyi yorumlamak ve anlamlı bilgi çıkartmak için XML okuyucuları ve yazıcıları standart bir eleman yapısına ihtiyaç duyarlar. Bu amaçla çeşitli sektörlerde verilerin tutulması için XML'den alt diller türetilmiştir. Standartlaşmayı sağlamak açısından bu sektörel alt diller de HTML gibi standart etiket adlarına sahiptir fakat işleyiş açısından XML'yi temel alır. World Wide Web Konsorsiyumu (W3C), finans, sağlık, sigorta, yayıncılık, trafik, emlak, seyahat, din ve akla gelebilecek her sektör için XML tabanlı ortak alt diller oluşturmaya devam etmektedir. Kurumlar birbirleri ile bilgi alışverişi yaparken, bu alt dillerden kendileri ile ilgili olanını alıp serbestçe kullanabileceklerdir.

XML ile E-Devlet uygulamalarında olduğu gibi farklı kurumların birbirleriyle olan iletişimi de daha kolay olacaktır. Veriler başka kurumlar tarafından kullanıldığında yeniden sorgulanabilir ve biçimlendirilebilir olacaktır. Örneğin, hastaneye giden bir vatandaşın tedavisi yapılmadan önce nüfus merkezinde tutulan veritabanından alınan veriler istemci program tarafından kolaylıkla anlaşılabilecektir. Internet bankacılığını kullanan birisi bağlı bulunduğu belediyenin veritabanını sorgulayarak vergisini öder duruma gelebilecektir. Bu tür uygulamaları gerçekleştirmek veri aktarımında ortak bir dilde birleşildiği için eskisine göre daha kolay olacaktır.

XML, HTML'nin yerini almak için üretilen bir dil değildir. Aksine HTML ile XML birlikte kullanılıp daha yetenekli veri sayfaları oluşturulabilir. XML'in kullanım alanı internet ile sınırlı değildir. Birbiriyle bilgi alışverişi yapması gereken uygulamalar için ortak bir dil olarak kullanılabilir. Böylelikle birbirinden bağımsız, tamamen ayrı sistemler XML aracılığıyla bilgi alış verişinde bulunabilirler. Paylaşmanız gereken verileri XML biçiminde dağıttığınızda verinizi alan bütün uygulamalar bilgiyi anlamlandırabilecektir.

XML belgesi içinde yer alan veriler, etiketlerle tanımlandıklarından hem bilgisayarlar hem de insanlar tarafından okunabilir. Tabii ki kimi karmaşık (kompleks) belgelerin anlaşılması zordur, uygulamalar tarafından okunabilir. İnternet üzerinde XML iki türlü kullanılabilir: İstemci tarafında ya da sunucu tarafında XML belgelerini işlemek

Sunucu tarafında işlenen XML verileri HTML biçiminde iletilirler. Bu, sunucu bilgisayarlara yük getiren ve genellikle önerilmeyen bir yöntemdir. Zaten bu şekilde yeniden tanımlanabilen veri aktarmak mümkün olmaz ve bu işlem veriyi XML olarak yayınlamamak, XML'yi amacına uygun kullanmamak anlamına gelir. Sunucu tarafında XML çalıştırmanın sebebi XML'i desteklemeyen eski web tarayıcılarda yaşanan sorunları önlemektir.

Sunucu, kendisine gelen istek vasıtasıyla istemcideki tarayıcının ne olduğunu anlayıp, XML biçimlemelerini kabul edip etmeyeceğine karar verir. İstemcinin tarayıcısı XML uyumlu ise istenilen sayfayı gönderir. Tarayıcı XML uyumlu değilse XML belgesini biçimleyip tarayıcının anlayabileceği, biçimlendirilmiş hazır kodlar gönderir.

XML uyumlu web tarayıcılar sunucudan gelen verileri alıp biçimlendirebilirler. İstemci bilgisayara XML belgesi ile birlikte biçim bilgilerinin bulunduğu bir başka dosya da gönderilebilir. Varsa bu biçim dosyasını (stil sayfası, stil şablonu) da beraberinde yükleyen

web tarayıcı programı biçim dosyasına bakarak XML belgesindeki verileri gösterir. XML belgesine, ziyaretçinin kendi seçebileceği bir stil sayfası uygulanabilir.

XML için standart biçimleme dili olarak XSL önerilmektedir. Her yeni teknoloji gibi XML'nin de geliştiriciler tarafından benimsemesi için eski kalıplara destek sunması gerekmektedir. Bu amaçla CSS biçim dosyalarını kullanarak XML verilerini biçimlemek mümkündür.

XML yeni diller oluşturmak için kullanılabilir. Örneğin, WML (Wireless Markup Language) XML'den türetilmiştir.

XML belgesi oluşturabilmek için Not Defteri gibi metin düzenleme programlarından biri kullanılabilir. Ayrıca sadece XML veri alanlarını düzenlemek için oluşturulmuş programlar da tercih edilebilir. Örneğin, XML Notepad programı Microsoft'un sitesinden ücretsiz olarak indirilebilir.

XML ile veriler daha fazla kullanıcının kullanımına açıktır. XML, donanım ve yazılımdan bağımsız olduğundan veriler standart HTML tarayıcılarının dışındaki yazılımlar tarafından da kullanılabilir. Diğer istemciler ve uygulamalar XML belgelerine erişebilirler.

Veri tabanlarında tutulan veriler, web ortamındaki bilgisayarlar arasında veri paylaşımı için sıklıkla XML verilerine dönüştürülür. İnternet sitesi kullanıcılarına veri tabanını erişmek için izin vermek güvenlik açısında bir risktir. Bu nedenle veri tabanındaki verileri XML verilerine dönüştürerek paylaşmak güvenlik açısından daha uygundur.

XML'in en yaygın kullanım alanlarından biri elektronik ticaret uygulamalarıdır. XML, tedarikçiler, müşteriler, iş ortakları vb. kurumlar arasında bilgi paylaşımında avantaj sağlar. XML ayrıca yayıncılık sektöründe de kullanışlıdır. XML olarak saklanan bilgi, her tür yayıncılık formatına kolaylıkla dönüştürülebilir.

XML'nin dünyadaki her uygulama tarafından tanınması için uygulama geliştiricileri W3C konsorsiyumunun kurallarına bağlı kalmaktadırlar.

.NET Framework, yoğun bir biçimde XML kullanır ve XML verileri üzerinde işlemler yapmak için zengin özellikler sunar. ASP.NET uygulamalarında XML kullanmaya hangi durumlarda ihtiyaç duyulabileceği aşağıda açıklanmıştır.

- Zaten XML olarak depolanmış verilerin kullanılması gerektiğinde. Bu durum XML kullanan bir uygulama ile veri alışverişinde bulunmak zorunda kalındığında ortaya çıkar.
- Gelecekte ihtiyaç duyulabilecek veri alışverişinde uyum sorunu yaşamamak amacıyla verileri depolamak için XML kullanmak istendiğinde. XML kullanıldığından gelecekte diğer uygulamalar bu veriyi okuyabilirler.
- XML'ye bağlı bir teknoloji kullanmak istendiğinde. Örneğin, XML'ye dayalı çeşitli standartlar kullanan web servisleri teknolojisi.

1.2. HTML ile XML Arasındaki Farklar

HTML veri görüntülemek için tasarlanmıştır ve verinin nasıl göründüğüne odaklanır. HTML dili bir belgenin formatlanması amacıyla daha önceden tanımlanmış bir etiket (tag) kümesine sahiptir. XML veri tanımlamak için tasarlanmıştır ve verinin ne olduğuna odaklanır. XML verinin nasıl görüneceğiyle ilgili veri içermez. XML belgelerinden elde edilen verilerin sayfadaki görünümünü ayarlamak için CSS, XSLT stil sayfaları kullanılır.

HTML dosyalarının yapısı ve HTML dosyaları oluşturmak için kullanılan etiketler (örneğin <br>, <i>) önceden tanımlanmıştır. XML belgesinin yapısını ve etiketlerini ise uygulama geliştiricisinin kendisi oluşturur. Personel ile ilgili verilerin bulunduğu yukarıdaki örnekte geliştirici olarak kendi etiketlerimizi kullandık.

XML, HTML'nin yerini alması için tasarlanmamıştır. XML, HTML'nin tamamlayıcısıdır. XML, verileri tanımlamak için kullanılırken HTML verileri biçimlendirmek ve görüntülemek için kullanılır.

XML verileri yapılandırmak, depolamak, göndermek için oluşturulmuştur. Yukarıdaki örnekte görüldüğü gibi XML'de veri kendi içinde bir yapıya sahiptir. Bu şekilde XML belgesinde tutulacaktır ve ihtiyaç duyulduğunda gönderilecektir.

Verileri görüntülemek için HTML kullanıldığında veriler HTML içinde tutulur. XML ile veriler ayrı XML belgelerinde tutulabilirler. Bu yöntemle verilerdeki herhangi bir değişiklikte HTML kodlarında değişiklik yapmak gerekmez. HTML sadece verileri istenilen yerleştiriliş düzeninde görüntülemek için kullanılır.

HTML'de bazı etiketler kapatılmadan kullanılmaktadır. Örneğin, <p> etiketi. XML'de ise tüm etiketler kapatılmalıdır.

XML'de etiketler büyük küçük harf duyarlıdır. <ad> ile <Ad> farklı etiketlerdir.
<ad> Zuhal </ad> DOĞRU
<Ad> Zuhal </ad> YANLIŞ

XML'de etiketler düzgün sırayla kapatılmalıdır.
<u>Malzeme listesi</u> YANLIŞ
<u>Malzeme listesi</u> DOĞRU

Aralık (space) tuşu, sekme (tab) tuşu ya da yeni satır tuşu ile elde edilen boşluğu ifade eden beyaz boşluk **(white space)** XML'de korunur. HTML birden fazla olan beyaz boşluk karakterini tek beyaz boşluğa indirir.

HTML	Merhaba Dünya
Çıktı	Merhaba Dünya

XML	Merhaba Dünya
Çıktı	Merhaba Dünya

HTML etiketlerindeki kimi hatalara rağmen HTML tarayıcıları belgeleri gösterir. XML belgelerindeki hatalarda ise XML uygulamalarının çalışması durur. HTML dosyalarının uzantısı html veya htm'dir. XML belgelerinin uzantısı xml'dir.

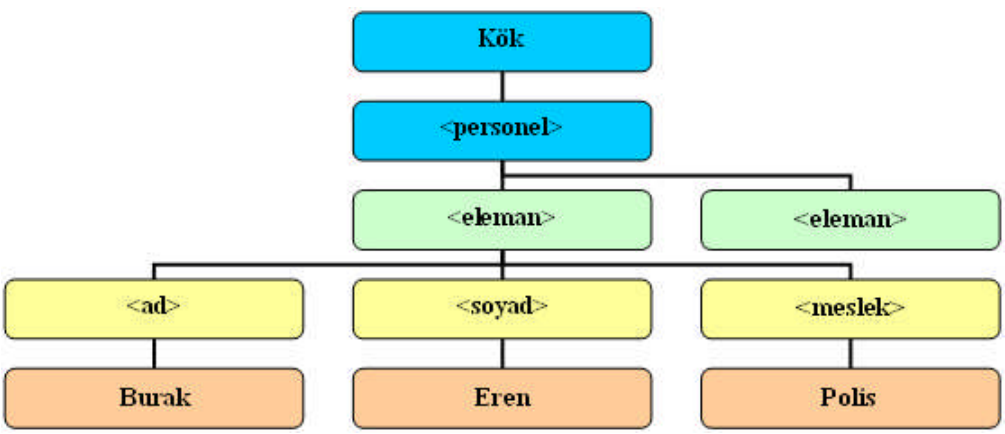
1.3. XML Söz Dizimi (Syntax) Kuralları

XML yazım kurallarını örnek bir XML belgesi üzerinden anlatalım. **File** menüsünden **New** komutu veriniz. Gelen pencerede **Category** kısmından **Basic Page** seçeneğini, **Basic Page** kısmından **XML** seçeneğini seçiniz

XML belgesinde verilerin aşağıdaki gibi dizilerek oluşturduğu görünüme **ağaç görünümü (tree view)** denir.

personel.xml

```
<?xml version="1.0" encoding="utf-8"?>
<personel>
   <eleman>
       <ad>Burak</ad>
       <soyad>Eren</soyad>
       <meslek>Polis</meslek>
    </eleman>
    <eleman>
       <ad>Yunus</ad>
       <soyad>Hızır</soyad>
       <meslek>Savcı</meslek>
    </eleman>
</personel>
```



Şekil 1.1: Personel.xml belgesinin ağaç yapısı

Personel.xml belgesinin ağaç yapısındaki her kutuya düğüm denir.

Resim 1.1: personel.xml belgesinin tarayıcıdaki görüntüsü

XML belgesinde etiketler ve etiketler arasında belirtilen veriler bulunmaktadır. Etiket ve veriden oluşun birime eleman (öge, element) adı verilir. Eleman isimleri aşağıda kurallara uymalıdır.

- İsimler harf, rakam ve diğer karakterleri içerebilir.
- İsimler rakam veya noktalama işaretleriyle başlayamaz.
- İsimler boşluk içeremez. Eğer birden fazla kelime kullanılmak isteniyorsa kelimeler arasındaki boşluk yerine alt çizgi kullanılabilir.
- "-" (tire), "**.**" (nokta), "**:**" (iki nokta üst üste) karakterlerini kullanmaktan kaçınılmalıdır. Bazı yazılımlar bu karakteri farklı anlamlarda yorumlayabilir.
- İsimler "xml" harfleriyle başlayamaz. (XML, xml vs.)

XML belgeleri sıklıkla kendisiyle ilişkili bir veri tabanına sahiptir. Bu açıdan veri tabanındaki isimlendirme kurallarını XML belgelerinde de kullanmak yararlı bir uygulamadır.

Belgenin ilk satırındaki bildirim XML sürüm (version) bilgisi ve kullanılacak dil kodlamasını tanımlar. Örnek belgede XML 1.0 sürümü ve "utf-8" dil kodlaması kullanılacağı belirtilmiştir. Bildirim XML elemanı olmadığından kapanış etiketi kullanılmamıştır.

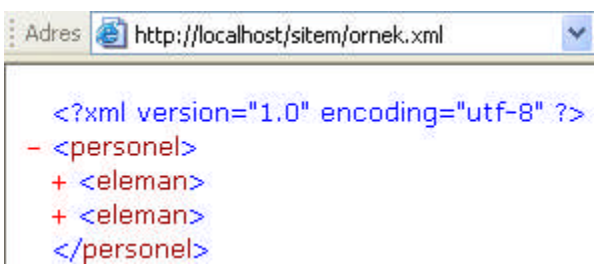
İkinci satır belgenin **personel** isimli **kök (root) elemanını** tanımlamaktadır. XML belgeleri sadece bir kök elemanına sahip olabilir. Kök eleman **yavru elemanları (child elements)** içinde barındırır. Yavru elemanlara sahip elemana **ana (parent) eleman** adı verilir. Tüm elemanlar yavru elemanlara sahip olabilir. Yavru elemanlar ana elemanlar içinde düzgün bir şekilde yerleştirilmelidir.

```
<kök eleman>
  <yavru eleman1>
    <alt yavru eleman1a>.....</alt yavru eleman1a>
    <alt yavru eleman1b>.....</alt yavru eleman1b>
  </yavru eleman2>
    <alt yavru eleman2a>.....</alt yavru eleman2a>
    <alt yavru eleman2b>.....</alt yavru eleman2b>
  </yavru eleman2>
</kök eleman>
```

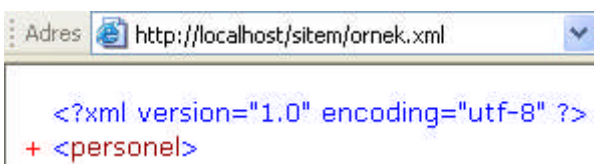
Örneğin üçüncü satırında **eleman** isimli alt eleman tanımlamaktadır.

NOT: Teknik bir terim olarak eleman kelimesi ile örnek XML belgesindeki eleman kelimesinin birbirinden farklı olduğuna dikkat ediniz. Dosyadaki eleman "bir kurumda personeli oluşturan insanlardan biri" anlamındadır.

Dördüncü, beşinci, altıncı satırlarda **ad, soyad, meslek** yavru elemanlar tanımlanmaktadır. Tarayıcıda elemanın solunda bulunan eksi ve artı işaretleri tıklanarak elemanın yapısı daraltılıp genişletilebilir. Elemanların aralarında olan ilişki, elemanların önündeki "-" (eksi) işareti ve her alt elemanın bağlı bulunduğu elemandan daha sağda yazılmasıyla belirtilmektedir. Eksi işaretine tıklatınca bir elemana ait alt elemanlar gösterimden kalkmaktadır. Eksi işareti artı şeklini almaktadır.



Resim 1.2: Genişletilmiş halde personel isimli kök eleman ve daraltılmış halde eleman isimli yavru eleman



Resim 1.3: Daraltılmış halde personel isimli kök eleman

Yedinci satırda **eleman** elemanının sonu **</eleman>** ifadesiyle tanımlanmıştır. 8-12.satırlar arasında yavru elemanlarıyla birlikte ayrı bir **eleman** elemanı tanımlanmıştır. 13.satırda **personel** elemanının sonu tanımlanmıştır.

UYGULAMA: Aşağıdaki değişiklikleri yapıp dosyayı tekrar çalıştırarak sonucu yorumlayınız.

a)7.satırı siliniz.
b)12.satırdaki </eleman> ifadesini <eleman> ifadesiyle değiştiriniz.
c)12.satırı </eleman> ifadesini </çalışan> ifadesiyle değiştiriniz.

XML'de açıklama satırı eklenmesi aşağıdaki şekilde yapılır.

<!-- Açıklama satırı -->

Örnek üzerinden gösterirsek şu şekilde bir açıklama satırı kullanılabilir.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Personel Bilgileri -->
……..
```

XML'de bazı karakterler özel anlam taşır. XML elemanları içinde "<" karakteri kullanıldığında hata oluşur. Çünkü ayrıştırıcı **(parser)** bu karakteri yeni bir elemanın başlangıcı olarak yorumlar. Bu tür özel karakterler yerine aşağıdaki tablodaki değerler kullanılır.

&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

XML'de özellik **(attribute)** değeri tek veya çift tırnak işareti içine alınmalıdır.
<not tarih="06/06/2006">

XML söz dizimi **(syntax)** kurallarına uyan XML belgelerine **iyi biçimli (well formed)** XML belgesi denir.

NOT: XML belgelerinin iyi biçimli olup olmadığını test etmek için o belgeyi tarayıcınızda açınız. Bir hata varsa tarayıcınız bu hatayı görüntüleyecektir.

Aşağıda kitaplar.xml ve haberler.xml isimli örnek XML belgeleri verilmiştir. Kitaplar.xml belgesi Etkileşimli Web Uygulamaları 3 modülündeki örnek veri tabanının kitaplar tablosuna benzer veri yapısına sahiptir.

UYGULAMA : Bu belgelerin ağaç yapısını çiziniz.

kitaplar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<kitaplar>
   <kitap>
      <nu>1</nu>
      <ad>Nutuk</ad>
      <yazar>M.Kemal Atatürk</yazar>
    <yayinevi>Kanarya</yayinevi>
   </kitap>
   <kitap>
      <nu>2</nu>
      <ad>Mesnevi'den Seçmeler</ad>
      <yazar>Mevlana</yazar>
    <yayinevi>Serçe</yayinevi>
   </kitap>
   <kitap>
      <nu>3</nu>
      <ad>Çalıkuşu</ad>
      <yazar>Reşat Nuri Güntekin</yazar>
    <yayinevi>Doğan</yayinevi>
   </kitap>
</kitaplar>
```

haberler.xml

(Haberler.xml belgesinde KAYNAK isimli bir özellik kullanılmıştır. Bu özellikte ağaç yapısında HABERLER ile aynı seviyedeki bir düğüm olarak gösterilir.)

```
<?xml version="1.0" encoding="utf-8"?>
<HABERLER KAYNAK="ABC Haber Ajansı">
   <HABER>
      <TARIH>1.1.2007</TARIH>
      <BASLIK>Liderlerin yılbaşı mesajları </BASLIK>
      <ICERIK>Yılbaşı dolayısıyla liderler birer mesaj
yayınladılar. <ICERIK>
   </HABER>
   <HABER>
      <TARIH>1.1.2007</TARIH>
      <BASLIK>Bütün Yurtta Kar Bekleniyor</BASLIK>
      <ICERIK>Meteoroloji İşleri Genel Müdürlüğü, yarın Türkiye
genelinde etkili kar ve sağanak yağış beklendiğini açıkladı. ICERIK>
   </HABER>
</HABERLER>
```

Kitaplar.xml belgesi yukarıdaki gibi yapılandırıldığı gibi geliştiricinin tercihine göre farklı şekilde de yapılandırılabilir. Örneğin, aşağıda kitabın birden fazla yazara sahip olduğu durumları da ele alan ve kimi elemanları özellik olarak belirleyen örnek verilmiştir.

```
<kitaplar>
   <kitap nu=”1”>
        <ad></ad>
        <yazarlar></yazarlar>
           <yazar></yazar>
           <yazar></yazar>
       <yayinevi> </yayinevi>
   </kitap>
<kitaplar>
```

1.4. XML Ad Alanları

XML yaygınlaştıkça farklı sektörler için, farklı amaçlarla birçok XML dili oluşturulmuştur. Bu diller benzer isimli elemanlar içerebilirler. Bu diller birlikte kullanıldığında eleman isimlerinin çakışmaması için XML ad alanları kullanılır. XML ad alanı standardının mantığı, her XML dilinin tüm elemanlarını benzersiz olacak şekilde tanımlayan kendine ait bir ad alanına sahip olmasıdır. Böylelikle bir elemanın hangi dile ait olduğu ayırt edilir.

Tüm XML ad alanları URI (Universal Resource Identifiers) kullanır. URI'lar (Evrensel Kaynak Tanılayıcı) genellikle web sayfası URL'sine benzerler. Örneğin, http://www.test.com. Ad alanları, web üzerindeki gerçek bir yerin URL adresi değildir. XML ad alanları için URI'lerin kullanılmasının sebebi onların daha benzersiz olmasıdır. Ad alanı URI olmak zorunda değildir ve herhangi bir metin de kullanılabilir. URI yerine URN (Uniform Resource Name) ve GUID (Globally Unique Iidentifier) kullanılabilir. Özetle, bir XML dilinin tüm dünyada tek olduğunu belirtmek için ad alanı kullanılır ve ad alanının seçiminde farklı isimlendirmeler kullanılabilir.

Bir elemanın belirli bir ad alanına ait olduğunu belirtmek için, başlangıç etiketine **xmlns** özelliğini eklenir ve ardından ad alanı yazılır. Örneğin, herhangi bir ad alanının parçası olmayan <kitaplar> elemanını **<kitaplar xmlns=”http://www.test.com”>** şeklinde yazmak onu **http://www.test.com** ad alanının bir parçası haline getirir. Bu şekilde bir ad alanı atandığında bu ad alanı tüm yavru elemanlar için varsayılan ad alanı olur. Aşağıdaki örnekte hem <kitaplar> hem diğer yavru elemanlar ilgili ad alanı içinde yer almaktadır.

```
<kitaplar xmlns=”http://www.test.com”>
   <kitap>
      <nu>1</nu>
      <ad>Nutuk</ad>
      <yazar>M.Kemal Atatürk</yazar>
    <yayinevi>Kanarya</yayinevi>
   </kitap>
</kitaplar>
```

XML belgesinin farklı bölümleri için farklı ad alanları kullanılabilir. Gerekli olduğu her yerde ad alanının ismine yazmak yerine ad alanı ön eki **(prefix)** kullanılır. Ad alanı ön ekleri, ad alanını göstermek için kısa karakter dizileridir.

1.5. XML Schema (XML Şema)

XML verilerinin farklı platformlar arasında alışverişinde verinin belirli kurallara göre yazılmış olması gerekmektedir. XML verilerinin belirli kurallara göre doğruluğunun tespitinde **DTD (Document Type Definitions), XDR (XML Data Reduced) ve XSD (XML Schema Definitions)** teknolojileri kullanılmaktadır.

Bu teknolojiler ile XML belgesinin yapısı tanımlanır. Yani XML belgesi içindeki elemanların uyması gereken kurallar belirlenir. Örneğin veri tipleri belirlenebilir, XML ağacının yapısı tanımlanabilir. Bu sayede verilerin tutarlılığı da sağlanmış olur.

Günümüzde, XML verilerinin doğrulanması için kullanılan en yaygın teknoloji XSD'dir. XML Schema Definition için sadece XML Schema ifadesi de kullanılır. XML şema, XML tabanlıdır ve W3C tarafından kullanımı tavsiye edilmektedir.

XML şema, belge içindeki elemanları, özellikleri, hangi elemanların yavru eleman olduğunu, yavru elemanlar arasındaki düzeni, sıralamayı, yavru eleman sayısını, elemanların boş olduğunu veya metin içerdiğini, eleman ve özelliklerin veri tiplerini, eleman ve özelliklerin varsayılan ve sabit değerleri gibi yapısal özellikleri tanımlar.

Sıklıkla bir uygulama tarafından oluşturulan XML belgesinin yapısı diğer bir uygulama tarafından gerekli olan yapıya uymaz. Bu durumda XSLT ile var olan XML veri yapısı diğer yapıya dönüştürülebilir.

İyi biçimli bir XML belgesi, şema **(schema)** kurallarına da uyuyorsa geçerli **(valid)** XML belgesi olarak isimlendirilir.

kitaplar.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.test.com"
xmlns="http://www.test.com"
elementFormDefault="qualified">

 <xsd:element name="kitaplar" type="kitaplarTip"/>
 <xsd:complexType name="kitaplarTip">
  <xsd:sequence maxOccurs="unbounded">
   <xsd:element name="kitap" type="kitapTip"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="kitapTip ">
  <xsd:sequence>
   <xsd:element name="nu" type="xsd:string"/>
   <xsd:element name="ad" type=" xsd:string"/>
   <xsd:element name="yazar"  type="xsd:string"/>
   <xsd:element name="yayinevi" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

<xsd:schema> elemanı her XML Schema'nın kök elemanıdır. **kitaplar.xsd** belgesinin kök elemanı aşağıdaki gibi verilmiştir.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.test.com"
xmlns="http://www.test.com"
elementFormDefault="qualified">
```

xmlns:xsd="http://www.w3.org/2001/XMLSchema" ifadesi**,** şema içinde kullanılan eleman ve veri tiplerinin **http://www.w3.org/2001/XMLSchema** ad alanından geldiğini gösterir. Bu ifade ayrıca bu ad alanından gelen eleman ve veri tiplerinin **xsd** ön ekini kullanması gerektiğini belirtir.

targetNamespace="http://www.test.com" ifadesi bu şema tarafından tanımlanan elemanların (kitaplar, kitap, nu, ad, yazar, yayinevi) **http://www.test.com** ad alanından geldiğini gösterir.

xmlns="http://www.test.com ise varsayılan ad alanını ifade eder.

<xsd:element name="kitaplar" type="kitaplarTip"/> satırıyla **kitaplar** adlı eleman **kitaplarTip** tipinde tanımlanmıştır. <**xsd:complexType…** satırıyla **kitaplarTip** tipi tanımlanmıştır. **<xsd:sequence…** satırı elemanların sırasını belirtmektedir. Bu örnekte sadece **kitap** isimli eleman vardır. **MaxOccurs="unbounded"** özelliği bu elemanın sınırsız sayıda kullanılabileceğini belirtir. **<xsd:element name="kitap"…** satırıyla **kitap** adlı eleman **kitapTip** tipinde tanımlanmıştır.

<**xsd:complexType name="kitapTip"**> satırıyla **kitapTip** tipi tanımlanmıştır. **<xsd:sequence…** satırıyla elemanların sırası belirtilmiştir. **<xsd:element…** satırlarıyla **nu, ad, yazar, yayinevi** adlı elemanlar **xsd:string** tipinde tanımlanmıştır.

Not : Diğer elemanları içerdiğinden kitaplar ve kitap elemanlarının tipi complexType'dır. Nu, ad, yazar, yayinevi elemanları alt elemana sahip olmadıklarından tipleri simpleType'dır.

İyi biçimli belgeler bile ciddi hatalar içerebilir. XML Schema kullanımı ile bu hataların çoğu bulunabilir.

1.6. XSL (Extensible Stylesheet Language)

XSL **(Extensible Stylesheet Language)**, W3C tarafından geliştirilmiş XML tabanlı stil sayfası dilidir. XSL, XML belgelerinin nasıl görüntüleneceğini tanımlar. XSL, üç kısımdan oluşur.

XPATH **(XML Path Language)** : XML belgelerinde gezinmek **(navigate)**, veri bulmak için kullanılan dildir. Diğer bir anlatımla XML ağacının içindeki düğümleri

adreslemeyi ve ağacın belirli kısımlarını adresleyerek ağaç içinde dolaşmamızı sağlayan dildir.

XSLT **(Extensible Stylesheet Language Transformations)** : XML belgelerini dönüştürmek **(trasform)** için kullanılan dildir.

XSL-FO **(XSL Formatting Objects)** : XML belgelerini biçimlendirmek **(format)** için kullanılan dildir.

1.7. XPATH (XML Adresleme Dili)

Xpath ile XML belgesi içindeki eleman, özellik gibi düğümler arasında dolaşmak mümkündür. XPath adres (yol) tanımlamaya benzer bir gösterim kullanır. Temel XPath sözdiminde kullanılabilecek ifadeler aşağıdaki tabloda verilmiştir.

İfade	**Anlamı**
/	XML belgesinin kökünü tanımlar. /kitaplar ifadesi <kitaplar> elemanını tanımlar. /kitaplar/kitap ifadesi <kitaplar> elemanın yavru elemanı olan her <kitap> elemanını seçer.
//	Nerede olduğuna bakmaksızın seçimle eşleşen yürürlükteki (current) düğümden itibaren belge içindeki düğümleri seçer.
@	Bir düğümün özelliğini seçer. /kitaplar/kitap/@id ifadesi <kitap> elemanın id isimli özelliğini seçer.
*	Yol tanımlamasındaki tüm yavru elemanları seçer. /kitaplar/kitap/* ifadesi <kitap> elemanın tüm yavru elemanlarını seçer.
.	Yürürlükteki düğümü gösterir.
..	Ana düğümü gösterir. Eğer yürürlükteki düğüm <ad> ise .. ifadesi <kitap> düğümünü gösterir.
[ ]	Bir seçim kriteri tanımlar. /kitaplar/kitap[yazar=’Mevlana’] ifadesi kritere uyan <yazar> elemanını içeren <kitap> elemanlarını seçer.
starts-with	Bir eleman metninin başlangıç karakterlerine bakarak elemanları seçer. /kitaplar/kitap[starts-with(yazar, 'R')] ifadesi R harfi ile başlayan metni içeren <yazar> elemanına sahip tüm <kitap> elemanlarını bulur.
position	Konuma bakarak elemanları seçer. /kitaplar/kitap[position()=2] ifadesi ikinci <kitap> elemanını seçer.
count	Belirtilen düğümün sayısını hesaplar. count(kitap) ifadesi <kitap> elemanlarının sayısını dönderir.

Tablo 1.1: Temel XPath Sözdizimi

Kriterleri birleştirmek için **and** anahtar kelimesi kullanılabilir.

1.8. XSLT (Extensible Stylesheet Language Transformations)

XML belgesini diğer XML belgesine veya diğer formatlara (HTML, XHTML) dönüştürmek için kullanılır. Kaynak belgeyi değiştirmeden bir hedef belge oluşturur. XML belgesini HTML belgesine dönüştürme işleminde XML belgesi kaynak belge, HTML belgesi hedef belgedir.

XML'de veri ile verinin sunumunun birbirinden ayrılması bir avantaj oluşturmaktadır. Bu avantaj XML verilerinin XSLT ile birleştirilerek dinamik olarak dönüşümün gerçekleştirilip herhangi bir formatta verilerin sunulabilmesidir. Yani veriler her zaman XML belgesinde tutulup, isteğe göre farklı şekillerde kullanıcıya sunulabilmektedir.

XSLT, bir XML belgesinin dönüşümü için ilk önce belgeyi ayrıştırarak (parse) bir ağaç yapısı oluşturur. Oluşan ağaç yapısının kök elemanından başlayarak ağacı tarar ve XSL stil dosyasında tanımlanan kurallara göre belgeyi dönüştürür. XSLT, dönüşüm işlemini her XML elemanını (X)HTML elemanına dönüştürerek gerçekleştirir. XSLT, ağaç yapısını tararken Xpath ifadelerini kullanarak ağacın farklı kesimlerine erişir.

Bir XML belgesini başka bir XML belgesine dönüşümüne örnek olarak cep telefonlarındaki uygulama verilebilir. Cep telefonlarındaki WAP uygulamalarında kullanılan **WML (Wireless Markup Language)** cep telefonlarındaki kısıtlamalar düşünülerek XML dilinde tanımlanmış bir işaretleme dilidir. XML belgeleri XSLT kullanılarak WML'e dönüştürülür ve cep telefonlarında görüntülenir.

XSLT ile çıktı dosyasına eleman ve özellik eklenebilir, çıkış dosyasından eleman ve özellik çıkartılabilir. Ayrıca elemanları sıralanabilir, tekrar düzenlenebilir, elamanlar gizlenebilir.

XSLT, XML belgelerinde veri aramak, gezinmek için XPath'i kullanır. Dönüşüm işleminde XSLT, şablonlarla eşleşmesi gereken kaynak belgenin kısımlarını XPath vasıtasıyla bularak tanımlar.

CSS'yle kıyaslandığında XSLT'nin kullanımı daha çok tavsiye edilir.

XSL belgelerini şablon olarak kullanmak için temelde iki yol vardır. Birincisi, XSL belgesinin içeriği HTML kalıplarıyla örülüp XML belgesini doğrudan görüntüleme, ikincisi HTML ya da ASP.NET belgesi içinde XSL ve XML işleyip sonucu yansıtma.

İkinci yöntemin kullanıldığı örnekler ilerleyen sayfalarda verilecektir. Şimdi birinci yöntemi uygulayarak bir örnek yapalım. Önce XML'yi HTML'ye dönüştürmek için kullanılan xsl dosyasını oluşturalım.

kitaplar1.xsl

```
     <?xml version="1.0" encoding="utf-8"?>
     <xsl:stylesheet                                        version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:template match="/">
     <html>
     <body>
     <table border="1">
     <tr bgcolor="#FFCCFF">
       <th>Sıra Nu</th>
       <th>Kitap Adı</th>
       <th>Yazar Adı</th>
       <th>Yayınevi</th>
     </tr>
     <xsl:for-each select="kitaplar/kitap">
     <tr>
     <td><xsl:value-of select="nu" /></td>
     <td><xsl:value-of select="ad" /></td>
     <td><xsl:value-of select="yazar" /></td>
     <td><xsl:value-of select="yayinevi" /></td>
     </tr>
         </xsl:for-each>
         </table>
       </body>
       </html>
       </xsl:template>
     </xsl:stylesheet>
```

XSL stil sayfası, şablon adı verilen bir veya daha fazla kurallar setinden oluşur. Eşleşen düğüme **(node)** uygulanacak kuralları içerir. <**xsl:template**> elemanı şablonları yapılandırmak için kullanılır. **Match** özelliği XML elemanı ile şablonu ilişkilendirmek için kullanılır. Match, tüm XML belgesi için bir şablon tanımlamak amacıyla da kullanılabilir. Match özelliğinin değeri bir XPath ifadesidir. Örnekteki **match="/"** ifadesi tüm belgeyi tanımlar. Diğer bir ifadeyle XML belgesinin kök elemanı ile şablonu ilişkilendirir. <**xsl:for-each**> elemanı XML belgesi içindeki elemanlara bir döngü içerisinde erişmemizi sağlar. <**xsl:value-of** > belirtilen elemana erişim sağlayarak değerini almayı mümkün kılar.

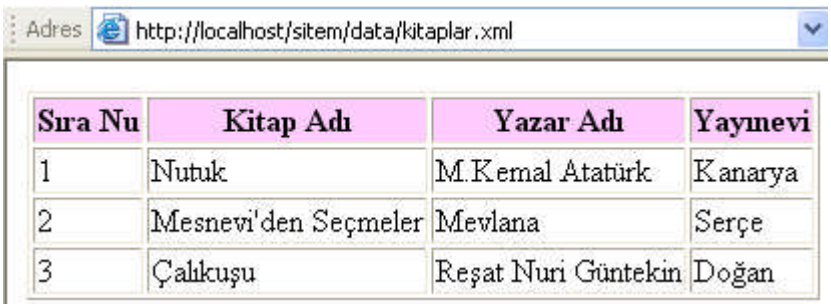
XSL stil sayfasının kendisi de bir XML belgesi olduğundan XML bildirimi ile başlamıştır : <**?xml version="1.0" encoding=" utf-8"?**>

<**xsl:stylesheet**> elemanı bu belgenin XSLT stil sayfası belgesi olduğunu belirtir. XSLT eleman ve özelliklerine erişmek için XSLT ad alanı ve versiyon numarası bildirilir. Sayfadaki **xmlns:xsl="http://www.w3.org/1999/XSL/Transform"** ifadesi W3C XSLT ad alanını belirtir. Belgenin sonundaki <**/xsl:stylesheet**> satırıyla bu eleman kapatılmıştır.

<**xsl:template**> içinde çıktıyı oluşturmak için HTML elemanları kullanılmıştır. <**xsl:template match="/"**> belgenin köküne uygulanacak şablonu tanımlamaktadır. Bu eleman <**/xsl:template**> satırıyla kapatılmıştır.

<xsl:for-each select="kitaplar/kitap"> satırı ile **kitap** düğümü içinde bir döngü oluşturulmuştur. **<xsl:value-of select=…>** satırlarıyla **nu, ad, yazar, yayinevi** elemanlarına erişim sağlanarak değerleri alınmıştır.

XML belgesinden stil sayfasına bağlanmak için **kitaplar.xml** belgesinin ikinci satırına **<?xml-stylesheet type="text/xsl" href="kitaplar1.xsl"?>** satırını ekleyelim. Ardından tarayıcıda görüntüleyelim.



Sıra Nu	Kitap Adı	Yazar Adı	Yayınevi
1	Nutuk	M.Kemal Atatürk	Kanarya
2	Mesnevi'den Seçmeler	Mevlana	Serçe
3	Çalıkuşu	Reşat Nuri Güntekin	Doğan

Resim 1.4: Kitaplar1.xsl ile dönüştürülmüş kitaplar.xml 'nin tarayıcıdaki görüntüsü

Görüldüğü gibi XML belgesinin sadece veri içermesine rağmen tarayıcıda html sayfasından farksız görüntülenmesinin nedeni, belgeye bağlanmış bir xsl belgesinin de yüklenip biçimlendirme için şablon olarak kullanılmasıdır.

NOT: Örneklerde kitaplar.xml belgesine satırlar eklenmektedir. Bu eklenen satırlar sadece o örnek içindir. Diğer örneklerde kitaplar.xml belgesinin özgün (orijinal) hali kullanılacaktır.

Örnek kitaplar.xml belgesi için farklı XSL belgeleri (kitaplar2.xsl, kitaplar3.xsl) kullanıp aynı verinin farklı görünümlerini içeren sayfalar elde edelim.

kitaplar2.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Kitaplar</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="kitap">
<p>
<xsl:apply-templates select="ad"/>
<xsl:apply-templates select="yazar"/>
</p>
</xsl:template>
```

```
<xsl:template match="ad">
Kitap adı : <span style="color:#ff0000">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
<xsl:template match="yazar">
Yazar adı : <span style="color:#0099FF">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
</xsl:stylesheet>
```

Resim 1.5: Kitaplar2.xsl ile dönüştürülmüş kitaplar.xml 'nin tarayıcıdaki görüntüsü

Kitaplar3.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<html                                          xsl:version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
      <body      style="font-family:Arial;font-size:12pt;background-
color:#CCFFCC">
        <xsl:for-each select="kitaplar/kitap">
          <div                                      style="background-
color:#66CCFF;color:white;padding:4px">
            <span            style="font-weight:bold"><xsl:value-of
select="nu"/></span>
             - <xsl:value-of select="ad"/>
          </div>
          <div       style="margin-left:20px;margin-bottom:1em;font-
size:10pt">
            <xsl:value-of select="yazar"/>
            <span style="font-style:italic">
              - <xsl:value-of select="yayinevi"/> Yayınevi
            </span>
          </div>
        </xsl:for-each>
      </body>
    </html>
```

Resim 1.6: Kitaplar3.xsl ile dönüştürülmüş kitaplar.xml 'nin tarayıcıdaki görüntüsü

1.9. XML DOM (Document Object Model)

DOM (Belge Nesne Modeli), XML belgesininin hafızada gösterimidir. DOM, bir belgeye kodlarla dinamik olarak erişmeyi ve bir belgenin stilini, yapısını, içeriğini güncellemeyi sağlayan platform ve dilden bağımsız bir arayüzdür **(interface)**. XML elemanlarını elde etmek, değiştirmek, eklemek veya silmek için bir standart tanımlar. DOM, uygulama ile XML belgeleri arasında bir köprü gibi çalışır. DOM, bir W3C standartıdır. 3 kısma ayrılır.

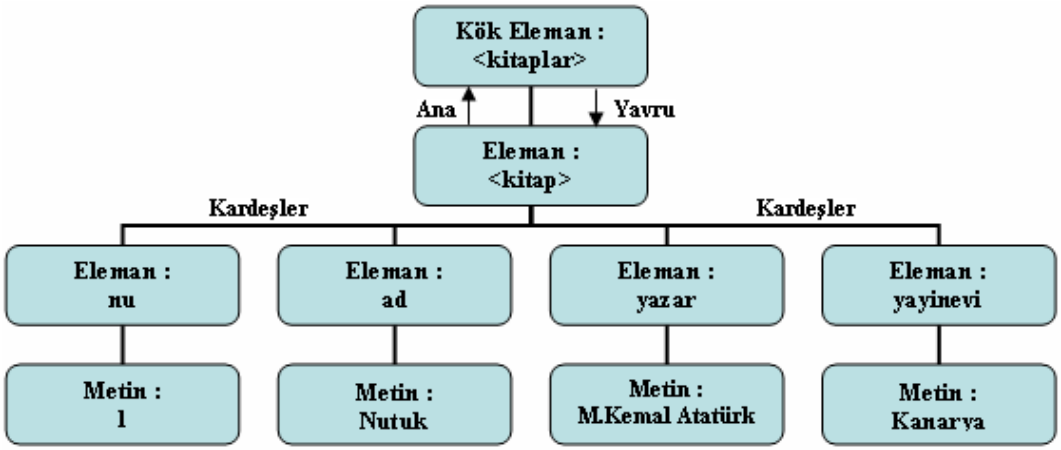
- Core DOM – Herhangi bir yapılandırılmış belge için standard model.
- XML DOM – XML belgeleri için standard model.
- HTML DOM – HTML belgeleri için standard model.

XML DOM, tüm XML nesneleri ile özelliklerini ve onlara erişmek için kullanılan metodları tanımlar. Diğer bir ifadeyle, DOM, XML'nin içindeki ögeleri nesne olarak ele alıp nesnelere ulaşmak için özellikler ve metotlar sunar. Uygulama geliştiricisi de kodlar vasıtasıyla bu özellik ve metotları kullanır.

DOM'da XML belgesindeki her şey bir düğümdür. DOM'a göre;

- Tüm belge, belge düğümüdür.
- Her XML elemanı bir eleman düğümüdür.
- XML belgesi içindeki metin bir metin düğümüdür.
- Her özellik bir özellik düğümüdür.
- Açıklamalar bir açıklama düğümüdür.

DOM, XML belgelerini düğüm olarak adlandırılan eleman, özellik, metin vs. düğümlerinden oluşan bir ağaç yapısı olarak sunar. Aşağıdaki şekilde her kutu XmlNode nesnesi olarak isimlendirilen bir düğümü gösterir.



Şekil 1.2: kitaplar.xml belgesinin düğüm ağacı olarak hafızada gösterimi

Metin değerleri (1, Nutuk, M.Kemal Atatürk, Kanarya) eleman düğümleri içinde değil metin düğümleri içinde saklanır. Örneğin, <ad>Nutuk<ad> satırında <ad> eleman düğümü "Nutuk" değerine sahip metin düğümünü tutar. Yani, "Nutuk" değeri <ad> elemanının değil metin düğümünün değeridir.

XML DOM, XML belgesini bir düğüm ağacı olarak görür. Ağaç içerisindeki tüm düğümler birbiriyle ilişki içerisindedir. Düğümlerin içeriği silinebilir, değiştirilebilir, düğüm ağacına yeni düğümler eklenebilir. Düğüm ağacı düğüm setleri ve onlar arasındaki ilişkileri gösterir. Ağaç kök elemandan başlar, ağacın en alt seviyesindeki metin düğümlerine kadar dallanır.

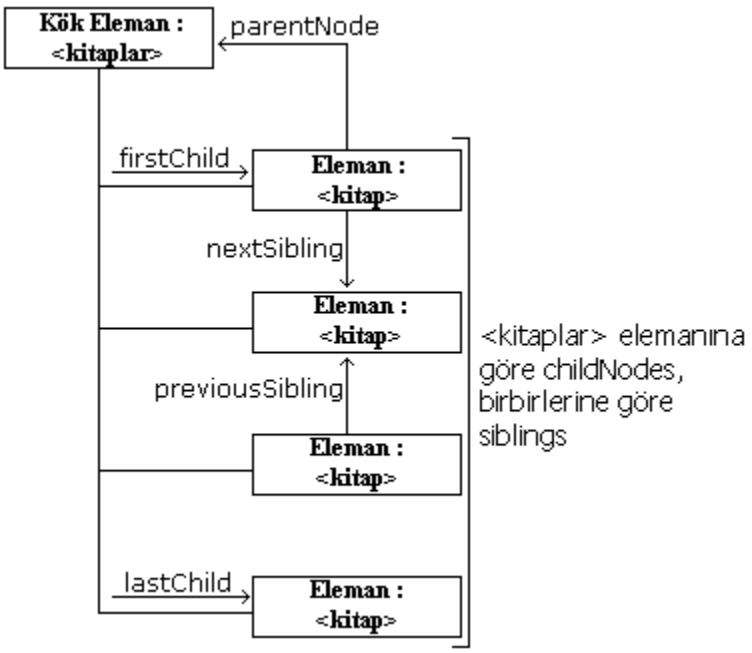
Düğüm ağacındaki düğümler birbirleriyle hiyerarşik (sıradüzensel) bir ilişki içerisindedir. Ana (**parent**), yavru **(child)** ve kardeş **(sibling)** kelimeleri bu ilişkiyi tanımlamak için kullanılır. Ana düğümler yavru düğümlere sahiptir. Aynı seviyedeki yavru düğümler kardeş düğümler olarak isimlendirilir.

Bazı XML DOM özellikleri (a'nın bir düğüm nesnesi olduğunu farz edelim.)

a.nodeName = a'nın ismi
a.nodeValue = a'nın değeri
a.parentNode = a'nın ana düğümü
a.childNodes = a'nın yavru düğümü
a.attributes = a'nın özellik düğümleri

XML DOM Metodları

a.getElementsByTagName(isim) = Belirtilen etiket ismine sahip tüm elemanları getirir.
a.appendChild(düğüm) = a'ya yavru düğüm ekler.
a.removeChild(düğüm) = a'dan yavru düğüm siler.



Şekil 1.3: Düğüm ağacının bir parçası ve düğümler arasındaki ilişki

1.10. XML Belgelerinin İşlenmesi

Uygulamaların XML belgelerini işlemesi için birçok yazılım geliştirilmiştir. Bu yazılımlara **XML processor (XML işlemcisi)** veya **XML parser (XML ayrıştırıcısı)** adı verilir.

Çoğu tarayıcı XML'yi okumak ve işlemek için yerleşik **(built-in)** XML ayrıştırıcısına **(parser)** sahiptir. Ayrıştırıcı, XML'yi okuyarak hafızaya alır ve onu program kodları tarafından erişilebilen XML DOM nesnelerine dönüştürür. Tarayıcılardaki ayrıştırıcılar birbirinden farklı olsalar da hepsi düğümlere erişmek, eklemek ve silmek için fonksiyonlara sahiptir.

1.11. XML Belgeleri Yazma

NET Framework'te **System.Xml** ad alanındaki sınıflar kullanılarak XML verisi üzerinde işlemler yapılabilir. .NET'te bir belgeye XML verisi yazmak için aşağıdaki yollar kullanılabilir.

- XMLDocument sınıfı kullanarak hafızada belge oluşturmak için DOM teknikleri kullanılabilir ve Save metodunu çağırarak belge bir dosyaya yazılabilir. XMLDocument XML'yi düğüm nesnelerinden oluşan ağaç yapısını kullanarak gösterir.
- XmlTextWriter kullanarak belge doğrudan bir katara **(stream)** yazılabilir. Bu işlem veriyi yazarken düğüm düğüm veri çıktısı üretir.

XML verisi üzerinde arama, dönüştürme, doğrulama işlemleri yapılacaksa XmlDocument kullanılmalıdır. XmlDocument, belgede herhangi bir yere yeni bir düğüm eklemeyi sağladığından doğrusal olmayan şekilde XML belgesi yazmayı mümkün kılar. XmlTextWriter, bir dosyaya doğrudan yazma açısından daha basit bir yöntem sunar.

NOT : Bir dosyada depolanmayan XML verisi oluşturmak için XmlDocument ve XmlTextWriter kullanılabilir. Her ikisi de herhangi bir katara veri yazılmasına izin verir. XmlDocument XML verisinin string veri olarak elde edilmesine olanak tanır. Bu teknikleri kullanarak XML belgesi yapılandırabilir ve farklı bir depolama yerine eklenebilir. Örneğin, veri tabanı tablosundaki metin tabanlı bir alana.

XmlTextWriter'ı kullanarak XML belgesi oluşturan bir örnek yapalım. Bu örnekte kitaplar.xml belgesinin bir benzeri **kitaplik.xml** adıyla oluşturulacaktır.

NOT: Örnek sayfalar verilirken sayfaları oluşturan tüm satırlar yerine sadece sayfanın üç farklı bölümü verilmiştir.

- Sayfanın başına eklenmesi gerekli import satırı
- <Script> blokları arasındaki kodlar
- Web formu oluşturan satırlar.

xmlBelgesiYazma.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(Src As Object, E As EventArgs)
Dim xmlBelge As String = Server.MapPath("kitaplik.xml")
Dim yazici As New XmlTextWriter(xmlBelge, Nothing)
yazici.Formatting = Formatting.Indented
yazici.Indentation = 3
yazici.WriteStartDocument()
yazici.WriteStartElement("kitaplar")

yazici.WriteStartElement("kitap")
yazici.WriteElementString("nu", "1")
yazici.WriteElementString("ad", "Nutuk")
yazici.WriteElementString("yazar", "M.Kemal Atatürk")
yazici.WriteElementString("yayınevi", "Kanarya")
yazici.WriteEndElement()

yazici.WriteStartElement("kitap")
yazici.WriteElementString("nu", "2")
yazici.WriteElementString("ad", " Mesnevi'den Seçmeler")
yazici.WriteElementString("yazar", " Mevlana")
yazici.WriteElementString("yayınevi", " Serçe")
yazici.WriteEndElement()

yazici.WriteEndElement()
yazici.Close()
End Sub
</script>
```

XmlTextWriter nesnesi yazılmak istenilen dosyanın fiziksel yolu belirtilerek oluşturulmuştur.

Formatting ve **indentation** özellikleriyle XML verisinin hiyerarşik yapıda otomatik olarak girintilendirilmesi belirtilmiştir. Girintilendirme için kullanılacak boşluk sayısı 3'tür. Girinti ekleme işlemi belgenin daha kolay okunması ve yorumlanması içindir.

WriteStartDocument metodu ile XML bildirimi (<?xml version="1.0"?>) yazılmıştır.

WriteStartElement("kitaplar") ile ana eleman oluşturulmuştur. Ardından yavru eleman olan <kitap> elemanı **WriteStartElement("kitap")** satırıyla oluşturulmuştur.

Sonraki satırlarda <kitap> elemanı içindeki kitap hakkında bilgiler içeren elemanlar eklenmiştir. Bu elemanların kendilerine ait yavru elemanları yoktur. Bu nedenle **WriteElementString** metoduyla bu elemanlar değerleriyle birlikte oluşturulmuşlardır.

Buraya kadar <kitap> elemanı için tüm veriler yazılmıştır. Son olarak **WriteEndElement** metodu <kitap> elemanı kapatılmıştır. Ardından diğer <kitap> elemanı oluşturulmuştur. Son **WriteEndElement** metodu ile <kitaplar> elemanı kapatılmıştır. **WriteEndElement** ile eleman ismi belirtmeye gerek yoktur. **WriteEndElement**, her çağrıldığında en son açılan elemanın kapanış etiketini otomatik oluşturacaktır.

Son olarak **Close** metoduyla XmlTextWriter kapatılmıştır. Sayfa çağrıldığında tarayıcıda herhangi bir şey görüntülenmeyecektir. Sayfa kitaplik.xml belgesini oluşturacaktır.

kitaplik.xml

```
<?xml version="1.0"?>
<kitaplar>
   <kitap>
      <nu>1</nu>
      <ad>Nutuk</ad>
      <yazar>M.Kemal Atatürk</yazar>
      <yayınevi>Kanarya</yayınevi>
   </kitap>
   <kitap>
      <nu>2</nu>
      <ad> Mesnevi'den Seçmeler</ad>
      <yazar> Mevlana</yazar>
      <yayınevi> Serçe</yayınevi>
   </kitap>
</kitaplar>
```

1.12. XML Belgelerini Okuma

Bir XML belgesini okumak ve içinde gezinmek (navigate) için çeşitli yollar vardır :

- XMLDocument kullanmak : XML belgesi XMLDocument sınıfı kullanılarak yüklenebilir. Bir dosyadan veya katardan veriyi yüklemek için Load metodu çağrıldığında tüm XML verisi hafızada tutulur. XMLDocument ile XML verisi üzerinde değişiklikler yapılıp tekrar dosyaya kaydedilebilir.

- XPathNavigator kullanmak : XML belgesi, XpathNavigator içine yüklenebilir. XpathNavigator, System.Xml.Xpath ad alanında bulunur. XMLDocument gibi XpathNavigator'da XML belgesini hafızada tutar. İleri arama özelliklerine sahiptir. XMLDocument'tan farklı olarak belgede düzenleme yapılıp kaydedilmesi özelliğine sahip değildir.

- XMLTextReader kullanmak : XMLTextReader sınıfı kullanılarak tek seferde bir düğüm şeklinde belge okunabilir. XMLTextReader, diğer yöntemlere kıyasla daha az sunucu kaynağı kullanır fakat belgeyi baştan sona doğru sırayla incelemeyi gerektirir.

XML belgesinden kitap listesinin yüklenmesini her üç yaklaşımla gerçekleştiren örnekler oluşturalım.

1.12.1. XML DOM Kullanma

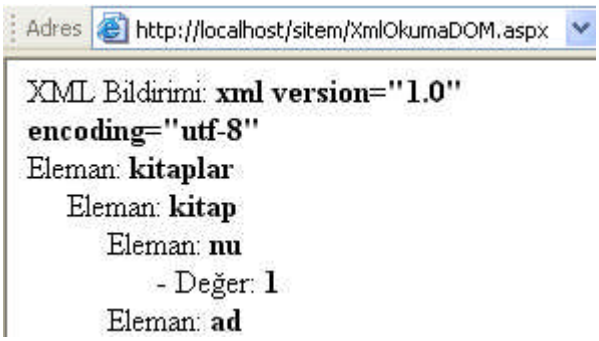
XMLDocument, verileri düğümler ağacı olarak depolar. Bir düğüm XML belgesinin temel ögesidir. Bir düğüm, eleman, özellik, açıklama veya bir eleman içindeki değer olabilir. Her düğüm ayrı bir **XmlNode** nesnesi tarafından temsil edilir. Düğümler koleksiyonlar içinde gruplandırılırlar.

XmlOkumaDOM.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">

Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim doc As XmlDocument = New XmlDocument()
doc.Load(xmlBelge)
xmlMetin.Text = YavruDugumDegerleriniAl(doc.ChildNodes, 0)
End Sub

Function YavruDugumDegerleriniAl(dugumListesi As XmlNodeList,
seviye As Integer) As String

Dim girinti As String = ""
Dim i As Integer=0

Do While i<seviye
```

```
girinti &= "     "
i += 1
Loop

Dim sb As New StringBuilder("")

For Each dugum As XmlNode In dugumListesi

Select Case dugum.NodeType
Case XmlNodeType.XmlDeclaration
sb.Append("XML Bildirimi: <b>")
sb.Append(dugum.Name)
sb.Append(" ")
sb.Append(dugum.Value)
sb.Append("</b><br />")
Case XmlNodeType.Element
sb.Append(girinti)
sb.Append("Eleman: <b>")
sb.Append(dugum.Name)
sb.Append("</b><br />")
Case XmlNodeType.Text
sb.Append(girinti)
sb.Append(" - Değer: <b>")
sb.Append(dugum.Value)
sb.Append("</b><br />")
Case XmlNodeType.Comment
sb.Append(girinti)
sb.Append("Açıklama: <b>")
sb.Append(dugum.Value)
sb.Append("</b><br />")
End Select

If dugum.Attributes IsNot Nothing Then
For Each ozellik As XmlAttribute In dugum.Attributes
sb.Append(girinti)
sb.Append(" - Özellik: <b>")
sb.Append(ozellik.Name)
sb.Append("</b> Değer: <b>")
sb.Append(ozellik.Value)
sb.Append("</b><br />")
Next
End If

If dugum.HasChildNodes Then
sb.Append(YavruDugumDegerleriniAl(dugum.ChildNodes, seviye+1))
End If

Next
Return sb.ToString()
End Function

</script>
```

```
<form runat="server" action="" method="post">
<asp:Literal ID="xmlMetin" runat="server" />
</form>
```



Resim 1.7: xmlOkumaDOM.aspx sayfasının tarayıcıdaki görüntüsü

Page_Load alt programında ilk olarak XMLDocument nesnesi oluşturulur. Bu nesnenin **Load** metodu ile xmlBelge isimli belgeden XML verisi yüklenir. Ardından **YavruDugumDegerleriniAl** isimli fonksiyonun değeri **xmlMetin** isimli **Literal** kontrolüne aktarılır. Literal kontrolü **<asp:Literal ID="xmlMetin" runat="server" />** satırıyla web form bölümünde oluşturulmuştur.

Page_Load alt programında **YavruDugumDegerleriniAl** fonksiyonu çağırıldığında fonksiyona parametre olarak belgenin yavru düğümleri (doc.childNodes) ve seviye olarak 0 değeri gönderilir. **YavruDugumDegerleriniAl** fonksiyondan geriye dönen string Literal kontrolle gösterilecektir. String tüm düğüm ve yavru düğümleri içerir. **XmlNodeList** tipindeki dugumlistesi, XmlNode nesnelerinin koleksiyonunu içerir.

YavruDugumDegerleriniAl fonksiyonu her girintilendirme seviyesi için kullanılacak olan üç boşluk karakterini içeren bir string oluşturur. Bu string, tarayıcıya yazılırken her satırın başına eklenir.

YavruDugumDegerleriniAl fonksiyonu, **For Each dugum As XmlNode In dugumListesi** satırıyla dugumlistesi isimli XmlNodeList içindeki tüm düğümler (node) arasında döngü oluşturur. Satırda **dugum As XmlNode** ifadesiyle **XmlNode** tipinde dugum nesnesi oluşturulmuştur. XmlNode nesnesi ögenin tipini (örneğin, açıklama, eleman, özellik, metin, ad, değer vb) tanımlayan NodeType gibi özelliklere sahiptir. **Select Case dugum.NodeType…End Select** bloğu arasında düğüm tipi kontrol edildikten sonra ilişkili veri stringe eklenmiştir. **XmlNodeType.XmlDeclaration** ifadesiyle düğümün bir XML bildirim düğümü olduğu, **XmlNodeType.Element** ifadesiyle düğümün bir eleman düğümü olduğu tespit edilmiştir.

Tüm düğüm tipleri isme (name) ve değere (value) sahip değildir. Örneğin, <ad> elemanının adı “ad” ama değeri boştur (empty) çünkü değeri takip eden metin (Text) düğümünde tutulur.

If dugum.Attributes IsNot Nothing Then satırıyla düğümün herhangi bir özelliğinin olup olmadığı kontrol edilmektedir. Bu işlem özellikler koleksiyonun boş olup olmadığını kontrol ederek yapar. For Each döngüsü kullanılarak özellikler işlenir.

Eğer bir düğüm yavru düğümlere sahipse **(If dugum.HasChildNodes Then) YavruDugumDegerleriniAl** fonksiyonu tekrar çağrılır. Bu fonksiyona bulunulan düğümün **ChildNodes** koleksiyonu ve girintileme seviyesi geçirilir. Tüm işlem bitirildiğinde en dıştaki For Each bloğu kapatılmış ve fonksiyon StringBuilder nesnesinin içeriğini geri döndürmüştür.

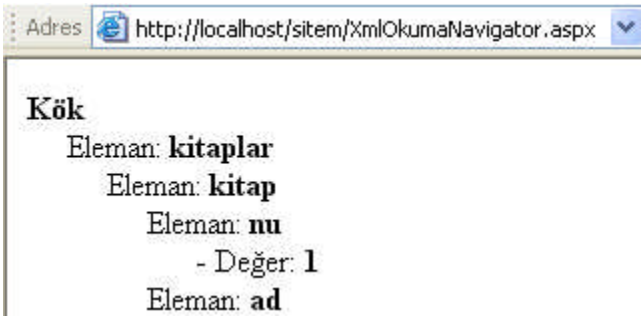
1.12.2. XPathNavigator Kullanma

XpathNavigator, XmlDocument sınıfına benzer şekilde çalışır. Tüm veriyi hafızaya yükler, ardından düğümler arasında hareket etmeyi sağlar. XML verisi içerisinde hareket etmek için **MoveToNext** gibi metodları vardır. XPathNavigator bir seferde sadece bir düğüme konumlandırılabilir. XmlDocument.CreateNavigator metodu kullanılarak XmlDocument'dan XPathNavigator oluşturulabilir.

XmlOkumaNavigator.aspx

```
<%@ Import NameSpace="System.Xml" %>
<%@ Import NameSpace="System.Xml.XPath" %>

<script runat="server">

Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim doc As New XmlDocument()
doc.Load(xmlBelge)
Dim xnav As XPathNavigator = doc.CreateNavigator()
xmlMetin.Text = XNavDegerleriniAl(xnav, 0)
End Sub

Function XNavDegerleriniAl(xnav As XPathNavigator, seviye As
Integer) As String

Dim girinti As String = ""
Dim i As Integer=0

Do While i<seviye
girinti &= "     "
i += 1
Loop

Dim sb As New StringBuilder("")

Select Case xnav.NodeType
Case XPathNodeType.Root
sb.Append("<b>Kök</b>")
sb.Append("<br />")
```

```
Case XPathNodeType.Element
sb.Append(girinti)
sb.Append("Eleman: <b>")
sb.Append(xnav.Name)
sb.Append("</b><br />")
Case XPathNodeType.Text
sb.Append(girinti)
sb.Append(" - Değer: <b>")
sb.Append(xnav.Value)
sb.Append("</b><br />")
Case XPathNodeType.Comment
sb.Append(girinti)
sb.Append("Açıklama: <b>")
sb.Append(xnav.Value)
sb.Append("</b><br />")
End Select

If xnav.HasAttributes Then
xnav.MoveToFirstAttribute()
Do
sb.Append(girinti)
sb.Append(" - Özellik: <b>")
sb.Append(xnav.Name)
sb.Append("</b> Değer: <b>")
sb.Append(xnav.Value)
sb.Append("</b><br />")
Loop While xnav.MoveToNextAttribute()
xnav.MoveToParent()
End If

If xnav.HasChildren Then
xnav.MoveToFirstChild()
Do
sb.Append(XNavDegerleriniAl(xnav, seviye+1))
Loop While xnav.MoveToNext()
xnav.MoveToParent()
End If

Return sb.ToString()
End Function

</script>
```



Resim 1.8: xmlOkumaNavigator.aspx sayfasının tarayıcıdaki görüntüsü

XnavDegerleriniAl fonksiyonu önceki örneğe benzer şekilde XML yapısını geri döndürür. XnavDegerleriniAl fonksiyonu bir düğüm koleksiyonuna değil tek bir düğüme konumlanan XPathNavigator nesnesini parametre olarak alır. Bu nedenle herhangi bir koleksiyon içinde döngü oluşturmanız gerekmez. Sadece aktif olan yani yürürlükteki düğümün verisi üzerinde işlem yapılabilir.

Önceki örnekteki **XmlNodeType** yerine **XPathNodeType** kullanılır. XPathNavigator'ın Nodetype **(xnav.NodeType)** özelliğinin değerleri önceki örnekteki XmlNode'un Nodetype özelliğinin değerleriyle neredeyse aynıdır. XPathNavigator'ın Nodetype özelliği XmlDeclaration düğüm tipini desteklemez.

Fonksiyon **If xnav.HasAttributes Then** satırını kullanarak yürürlükteki düğümün herhangi bir özelliği olup olmadığını kontrol eder. Varsa, **MoveToFirstAttribute** metodu ile ilk özelliğe hareket eder. **MoveToNextAttribute** metodu **False** olana kadar tüm özelliklere döngüyle ulaşır. False değerine ulaşıldığında döngüden çıkılır. Bu aşamada **xnav.MoveToParent()** satırıyla nesne tarafından başvurulan düğüm olan ana düğüme dönülür.

Fonksiyon **If xnav.HasChildren Then** satırını kullanarak yavru düğümlerinin var olup olmadığını tespit eder. Varsa **MoveToFirstChild** metodu ile ilk yavru düğüme gider. **MoveToNext** False değeri döndürene kadar kendini çağırır. Böylelikle tüm yavru düğümler için benzer işlemi gerçekleştirmiş olur. **MoveToNext** False değeri döndürdüğünde ana düğüme geri döner.

1.12.3. XML Belgesinde Arama

Bazen tüm XML belgesini işlemeye gerek yoktur. Sadece belgedeki bir bölümü işlemek gerektiğinde üzerinde işlem yapılacak bölümün aranıp bulunması gerekir. Böyle bir durumda XmlDocument'ın GetElementsbyTagName metodunu, Xpath'i veya XmlTextReader'ı kullanarak Xml belgesinde arama gerçekleştirilebilir

1.12.3.1. GetElementsbyTagName Metodunu Kullanma

Eğer eleman ismi biliniyorsa **XmlDocument.GetElementsByTagName** metodu kullanılabilir. Bu metod tüm belgeyi arar ve tüm uyan (eşleşen) XmlNode nesneleri içeren XmlNodeList nesnesini geri döndürür.

GetElementsbyTagName.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
' XML belgesini yükle.
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim doc As New XmlDocument()
doc.Load(xmlBelge)
' Belge içinde herhangi bir yerdeki tüm <ad> elemanını bul.
Dim sb As New StringBuilder()
Dim dugumListesi As XmlNodeList =
doc.GetElementsByTagName("ad")
For Each dugum As XmlNode In dugumListesi
sb.Append("Bulunulan: <b>")
' <ad> elemanın içerdiği metni göster.
sb.Append(dugum.ChildNodes(0).Value)
sb.Append("</b><br />")
Next
xmlMetin.Text = sb.ToString()
End Sub
</script>
```

Resim 1.9: GetElementsbyTagName.aspx sayfasının tarayıcıdaki görüntüsü

XmlElement.GetElementsByTagName metodu kullanılarak belirli eleman üzerinde XML belgelerinin bölümleri de araştırılabilir. Bu durumda XmlDocument eşleşmeyi bulmak için tüm düğümleri arar. Bu metodu kullanmak için önce bir elemanın karşılığı olan XmlNode nesnesini elde edilir ardından bu nesne XmlElement nesnesine dönüştürülür. Bu yöntemi kullanarak belirli bir kitabın yazarını bulan bir örnek yapalım.

GetElementsbyTagName2.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim doc As New XmlDocument()
doc.Load(xmlBelge)
Dim sb As New StringBuilder()
Dim dugumListesi As XmlNodeList =
doc.GetElementsByTagName("ad")
For Each dugum As XmlNode In dugumlistesi
sb.Append("Bulunulan: <b>")
Dim kitap_adi As String = dugum.ChildNodes(0).Value
sb.Append(kitap_adi)
sb.Append("</b><br />")
If kitap_adi = "Mesnevi'den Seçmeler" Then
' Sadece bu kitabın yazarını bul.
' İlk önce ana düğüm olan <kitap> elemanını elde et.
Dim ana As XmlNode = dugum.ParentNode
' Sonra ağaçta arama yap.
Dim yavruDugumListesi As XmlNodeList = (CType(ana,
XmlElement)).GetElementsByTagName("yazar")
For Each yavruDugum As XmlNode In yavruDugumListesi
sb.Append("   Bulunulan yazar: ")
sb.Append(yavruDugum.ChildNodes(0).Value)
sb.Append("<br />")
Next yavruDugum
End If
Next
xmlMetin.Text = sb.ToString()
End Sub
</script>
```

Resim 1.10: GetElementsbyTagName2.aspx sayfasının tarayıcıdaki görüntüsü

1.12.3.2. XPath ile Xml Belgesinde Arama

GetElementsByTagName metodu sınırlı özelliklere sahiptir. Sadece eleman ismine bağlı aramaya izin verir. Bir elemanın veya özelliğin değeri gibi bir kritere bağlı filtreleme işlemi yapamaz. Xpath, bir belgenin bölümlerinin elde edilmesini sağlar.

.NET'te XPath ifadesini çalıştırmak için **XPathNavigator**'ın **Select** metodu kullanılabilir veya **XmlDocument** sınıfının **SelectNodes** veya **SelectSingleNode** metodu kullanılabilir. Xpath'in kullanıldığı bir örnek yapalım.

XPathArama.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim doc As New XmlDocument()
doc.Load(xmlBelge)
Dim dugumListesi As XmlNodeList =
doc.SelectNodes("/kitaplar/kitap/ad")
' Kitap adlarını görüntüle.
Dim sb As New StringBuilder()
For Each dugum As XmlNode In dugumListesi
sb.Append("Bulunulan: <b>")
' Bu <ad> elemanı içindeki metni göster.
sb.Append(dugum.ChildNodes(0).Value)
sb.Append("</b><br />")
Next
XmlMetin.Text = sb.ToString()
End Sub
</script>
```

Resim 1.11: XPathArama.aspx sayfasının tarayıcıdaki görüntüsü

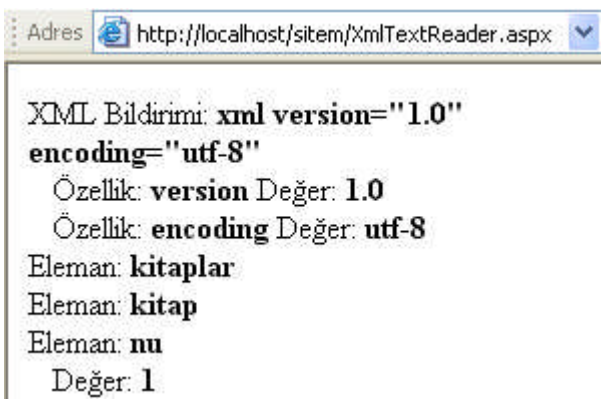
1.12.4. XmlTextReader Nesnesini Kullanma

XmlTextReader nesnesi ile XML belgesini okuma en basit yaklaşımdır, fakat bu yöntem en az esnekliği sağlar. Belge içeriği sırayla okunur. XmlDocument ve XPathNavigator ile yapıldığı gibi istenilen düğüme ulaşılamaz.

XmlTextReader.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
' Okuyucuyu oluştur.
Dim okuyucu As New XmlTextReader(xmlBelge)
Dim sb As StringBuilder = New StringBuilder()
' Tüm düğümler için döngü oluştur.
Do While okuyucu.Read()
```

```
Select Case okuyucu.NodeType
Case XmlNodeType.XmlDeclaration
sb.Append("XML Bildirimi: <b>")
sb.Append(okuyucu.Name)
sb.Append(" ")
sb.Append(okuyucu.Value)
sb.Append("</b><br />")
Case XmlNodeType.Element
sb.Append("Eleman: <b>")
sb.Append(okuyucu.Name)
sb.Append("</b><br />")
Case XmlNodeType.Text
sb.Append("   Değer: <b>")
sb.Append(okuyucu.Value)
sb.Append("</b><br />")
End Select
If okuyucu.AttributeCount > 0 Then
Do While okuyucu.MoveToNextAttribute()
sb.Append("   Özellik: <b>")
sb.Append(okuyucu.Name)
sb.Append("</b> Değer: <b>")
sb.Append(okuyucu.Value)
sb.Append("</b><br />")
Loop
End If
Loop
' Okuyucuyu kapat ve metni göster.
okuyucu.Close()
xmlMetin.Text = sb.ToString()
End Sub
</script>
```



Resim 1.12: XmlTextReader.aspx sayfasının tarayıcıdaki görüntüsü

XmlTextReader nesnesi içine kaynak dosya yüklenir. Ardından tek seferde tek düğüme hareket edilecek şekilde döngü başlar. Bir düğümden diğerine geçmek için XmlTextReader.Read metodu çağırılır. Bu metod son düğümü geçene kadar True değeri dönderir. Son düğümü geçince False değerini dönderir.

Aranılan düğüm tipi bulunduktan sonra sonraki adım yürürlükteki düğümün özelliği olup olmadığı kontrol etmektir. XmlTextReader özellikler koleksiyonuna sahip değildir. **AttributeCount** özelliği özelliklerin sayısını verir. **MoveToNextAttribute** False değerini döndürene kadar imleç bir sonraki özellik düğümüne ilerletilir.

Alt programın son iki satırında okuyucu kapatılır, ardından içerik Literal kontrolü kullanılarak yazdırılır. XmlTextReader kullanımında işlem bittiğinde okuyucuyu kapatmak zorunludur. Örnek sayfanın çıktısı daha önce XmlDocument ve XPathNavigator ile yapılan örneklerin çıktılarıyla oldukça benzerdir.

Beklenilen yapı bilindiğinde XmlTextReader'ın diğer yöntemlere göre daha hızlı ve daha güvenli Xml okumaya yardımcı olan ek metotları vardır. Örneğin, **MoveToContent** metodu. Bu metot, açıklama, beyaz boşluk, XML bildirimi gibi düğümleri atlayarak sonraki elemanın bildiriminde durur.

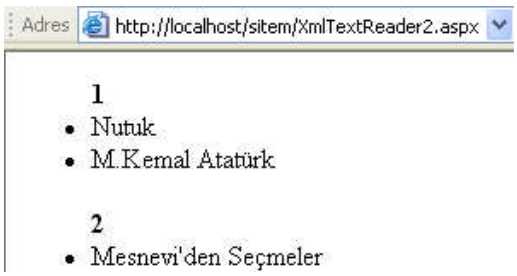
ReadStartElement metodu elemanın başlangıç etiketini okur ve aynı zamanda temel düzeyde geçerlilik (validation) kontrolü yapar. ReadStartElement çağrıldığında belgede bir sonra görüntülenmesi beklenilen elemanın ismi belirtilir. XmlTextReader, MoveToContent metodunu çağırır, ardından yürürlükteki elemanının belirtilen isme sahip olup olmadığını kontrol eder. Eğer yürürlükteki eleman belirtilen isme sahip değilse istisnai durum (exception) oluşur. Beyaz boşlukları atlamak ve elemanın kapanış etiketini okumak amacıyla ReadEndElement metodu kullanılabilir.

Eğer sadece metin içeren eleman okunmak istenirse, **ReadElementString** metoduyla eleman ismi belirtilerek başlangıç etiketi, içerik ve bitiş etiketine hareket edilebilir. Elde edilmek istenen veri string olarak döner. Bu yaklaşımı kullanarak XML veri çekmekle ilgili bir örnek yapalım.

XmlTextReader2.aspx

```
<%@ Import NameSpace="System.Xml" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim okuyucu As New XmlTextReader(xmlBelge)
Dim sb As New StringBuilder()
okuyucu.ReadStartElement("kitaplar")
' Tüm <kitap> elemanları oku.
Do While okuyucu.Read()
If (okuyucu.Name = "kitap") AndAlso _
(okuyucu.NodeType = XmlNodeType.Element) Then
okuyucu.ReadStartElement("kitap")
sb.Append("<ul><b>")
sb.Append(okuyucu.ReadElementString("nu"))
sb.Append("</b><li>")
sb.Append(okuyucu.ReadElementString("ad"))
sb.Append("</li><li>")
sb.Append(okuyucu.ReadElementString("yazar"))
```

```
sb.Append("</li></ul>")
End If
Loop
okuyucu.Close()
XmlMetin.Text = sb.ToString()
End Sub
</script>
```



Resim 1.13: XmlTextReader.aspx sayfasının tarayıcıdaki görüntüsü

1.13. XSL ile XML Verilerini Görüntüleme

ASP.NET'te XSL kullanılarak XML verilerini görüntülemek için **XslCompiledTransform** sınıfı veya **XML** kontrolü kullanılabilir.

1.13.1. XslCompiledTransform Kullanma

System.Xml.Xsl ad alanında bulunan XsCompiledlTransform sınıfı kullanılarak XSL ile XML verileri biçimlendirilmiş HTML'ye dönüştürülebilir. Örneğimizde kitaplar.xml belgesini daha önceki örneklerde oluşturulan kitaplar1.xsl sayfası ile kitaplar.htm adlı yeni bir dosyasına dönüştürelim.

XslCompiledTransform.aspx

```
<%@ Import NameSpace="System.Xml.Xsl" %>

<script runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
Dim xslBelge As String = Server.MapPath("data\kitaplar1.xsl")
Dim xmlBelge As String = Server.MapPath("data\kitaplar.xml")
Dim htmlBelge As String = Server.MapPath("kitaplar.htm")
Dim donustur As New XslCompiledTransform()
donustur.Load(xslBelge)
donustur.Transform(xmlBelge, htmlBelge)
End Sub
</script>
```

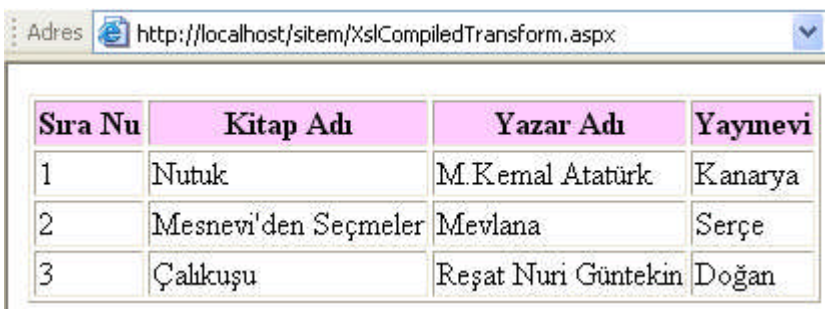
Sayfa çağrıldığında tarayıcıda herhangi bir şey görüntülenmeyecektir. Sayfa kitaplar.htm belgesini oluşturacaktır.

kitaplar.htm

```
<html>
  <body>
    <table border="1">
      <tr bgcolor="#FFCCFF">
        <th>Sıra Nu</th>
        <th>Kitap Adı</th>
        <th>Yazar Adı</th>
        <th>Yayınevi</th>
      </tr>
      <tr>
        <td>1</td>
        <td>Nutuk</td>
        <td>M.Kemal Atatürk</td>
        <td>Kanarya</td>
      </tr>
….
```

İstenirse aşağıda satır kullanılarak çıktı tarayıcıda görüntülenebilir.

donustur.Transform(xmlBelge,Nothing,Response.OutputStream)



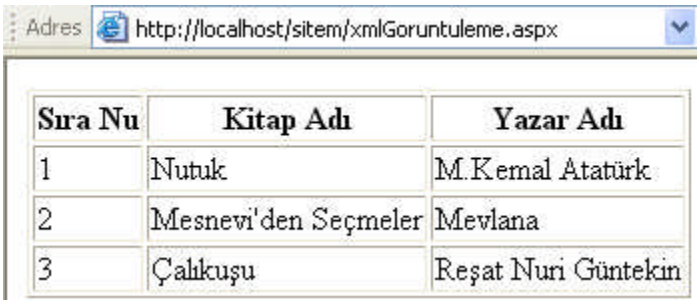
Resim 1.14: XslCompiledTransform.aspx sayfasının tarayıcıdaki görüntüsü

1.13.2. XML Kontrolünü Kullanma

XML kontrolü kullanılarak dönüştürülmüş HTML çıktısı sayfada görüntülenebilir. Xml kontrolü sayfanın bir bölümünde XLS dönüşümünün sonucunu görüntüler. Xml kontrolünde **DocumentSource** özelliği için xml belgesi, **TransformSource** özelliği için xsl belgesi belirtilir.

xmlGoruntuleme.aspx

```
<form runat="server" name="form1" method="post" action="">
<asp:Xml DocumentSource="data\kitaplar.xml" ID="xml1"
runat="server" TransformSource="data\kitaplar.xsl" />
</form>
```



Resim 1.15: xmlGoruntuleme.aspx sayfasının tarayıcıdaki görüntüsü

DocumentSource özelliği XML belgesinin sunucudaki adresini belirtir. TransformSource özelliği XML verilerine uygulanacak XSL stil sayfası belgesinin adresini belirtir.

Metin kutusuna girilen kitap adını kitaplar.xml belgesinde arayan, bulursa o kitapla ilgili ayrıntıları Xml kontrolünü kullanarak görüntüleyen bir örnek yapalım. Kitaplar.xml belgesini dönüştürmek için kitaplar5.xsl belgesi kullanılacaktır.

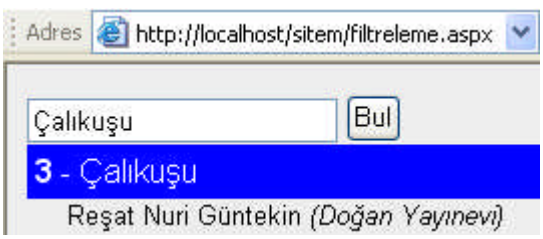
Filtreleme.aspx

```
<%@ Import Namespace="System.Xml.Xsl" %>

<script runat="server">
Sub Button1_Click(Src As Object, E As EventArgs)
  Dim arguman_liste As New XsltArgumentList()
  arguman_liste.AddParam("ad", "", TextBox1.Text)
  Xml1.TransformArgumentList = arguman_liste
  Xml1.Visible = True
End Sub
</script>

<form runat="server" name="form1" method="post" action="">
<asp:TextBox ID="TextBox1" runat="server" />
<asp:Button      ID="Button1"      runat="server"      Text="Bul"
OnClick="Button1_Click" />
<asp:Xml DocumentSource="data\kitaplar.xml" ID="xml1"
runat="server" TransformSource="data\kitaplar5.xsl" />
</form>
```

kitaplar5.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:param name="ad"></xsl:param>
  <xsl:template match="/">
    <HTML>
      <BODY  STYLE="font-family:Arial,  helvetica,  sans-serif;
font-size:12pt;
```

```
                background-color:#EEEEEE">
              <xsl:for-each select="kitaplar/kitap[ad = $ad]">
                <DIV    STYLE="background-color:blue;    color:white;
padding:4px">
                <SPAN                        STYLE="font-weight:bold;
color:white"><xsl:value-of select="nu"/></SPAN>
                - <xsl:value-of select="ad"/>
              </DIV>
              <DIV    STYLE="margin-left:20px;    margin-bottom:1em;
font-size:10pt">
                <xsl:value-of select="yazar"/>
                <SPAN STYLE="font-style:italic">
                  (<xsl:value-of select="yayinevi"/> Yayınevi)
                </SPAN>
              </DIV>
            </xsl:for-each>
          </BODY>
        </HTML>
      </xsl:template>
    </xsl:stylesheet>
```



Resim 1.16: Ekran çıktısı

1.14. XML Veri Bağlama

XML okumak, yazmak ve görüntülemek için uygulanan çeşitli yöntemlerden biri de **XmlDataSource** kontrolüdür. Bu kontrol elle kod yazımını belli ölçüde azaltmaktadır. XmlDataSource kontrolü AccessDataSource kontrolüne benzer şekilde çalışır. XmlDataSource ile XML belgelerine bağlanılır.

1.14.1. Hiyerarşik Olmayan Bağlama

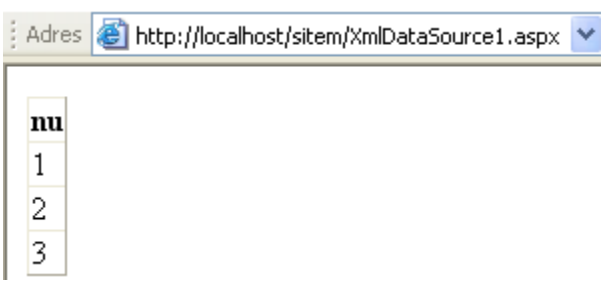
Hiyerarşik yapıda olan XML verisini Gridview gibi bir kontrolle görüntülendiğinde veriler hiyerarşik olmayan bir yapıda görüntülenir. Bu işlemi gerçekleştirmek için XmlDataSource kontrolünün veri dosyası **(DataFile)** olarak xml belgesi gösterilir. Ardından Gridview kontrolünün veri kaynağı olarak **(DataSourceID)** XmlDataSource kontrolü belirtilir. Örnekte kitabevi.xml belgesi veri dosyası olarak kullanılmıştır. Bu belgenin kitaplar.xml belgesinden farkı, <nu> elemanının <kitap> elemanının bir özelliği olarak ve <yazar> elemanının <yazarlar> elemanının yavru elemanı olarak kullanılmasıdır.

XmlDataSource1.aspx

```
<%@ Import NameSpace="System.Xml" %>

<form runat="server" action="" method="post">
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="True" DataSourceID="XmlDataSource1">
</asp:GridView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml" />
</form>
```

kitabevi.xml

```
<?xml version="1.0" encoding="utf-8"?>
<kitaplar>
   <kitap nu="1">
      <ad>Nutuk</ad>
      <yazarlar>
       <yazar>M.Kemal Atatürk</yazar>
      </yazarlar>
   <yayinevi>Kanarya</yayinevi>
   </kitap>
   <kitap nu="2">
      <ad>Mesnevi'den Seçmeler</ad>
      <yazarlar>
       <yazar>Mevlana</yazar>
      </yazarlar>
   <yayinevi>Serçe</yayinevi>
   </kitap>
   <kitap nu="3">
      <ad>Çalıkuşu</ad>
      <yazarlar>
       <yazar>Reşat Nuri Güntekin</yazar>
      </yazarlar>
   <yayinevi>Doğan</yayinevi>
   </kitap>
</kitaplar>
```

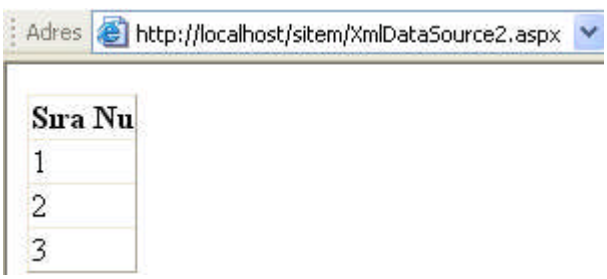


Resim 1.17: XmlDataSource1.aspx sayfasının tarayıcıdaki görüntüsü

Sayfa çağrıldığında XmlDataSource, kitabevi.xml belgesinden verileri çekip GridView'e XmlDocument nesnesi olarak sunar ve DataBind metodunu çağırır. GridView, **XmlDocument.Nodes** koleksiyonu içindeki her **XmlNode** için tüm özellikleri alır ve görüntüler. XmlDocument.Nodes koleksiyonu sadece ilk seviye düğümleri içerdiğinden (<kitap> elemanı ilk seviyeyi oluşturmaktadır.) sadece <kitap> elemanının özellikleri görüntülenmiştir. Bu düğümlerin her biri **XmlNode.Nodes** koleksiyonu içinde tutulan yuvalanmış düğümleri içerebilir. XmlDocument, XmlNode nesnelerinin sadece en üst seviyesindekilere eriştiğinden bu yuvalanmış düğümler görüntülenemez.

NOT: Örnekte web form bölümündeki bildirim ile sonuç elde edilmiştir. Bu işlem için kodlar da kullanılabilir. XmlDataSource nesnesi üzerinde GetXmlDocument metodu çağrıldığında XmlDataSource, XmlDocument nesnesi olarak belgenin içeriğini dönderir.

```
<form runat="server" action="" method="post">
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataSourceID="XmlDataSource1">
<Columns>
<asp:BoundField DataField="nu" HeaderText="Sıra Nu"
SortExpression="nu" />
</Columns>
</asp:GridView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml" />
</form>
```



Resim 1.18: XmlDataSource2.aspx sayfasının tarayıcıdaki görüntüsü

Bu durumda XML belgesi içindeki diğer veriler nasıl gösterilecektir? Bu sorunun çözümü için aşağıdaki yöntemlerden biri kullanılabilir.

- Elemanları filtrelemek için Xpath kullanılabilir.
- XML belgesini istenilen yapıya dönüştürmek için XSL dönüşümleri kullanılabilir.
- Bir veri kontrolünün içine diğer bir veri kontrolü yerleştirilebilir.
- Hiyerarşik veriyi destekleyen bir kontrol kullanılabilir. .NET'te bu amaca uygun TreeView kontrolü vardır.

1.14.1.1. Xpath Bağlama İfadelerini Kullanma

Xpath veri bağlama ifadeleri kullanılarak yuvalanmış elemanların metinleri görüntülenebilir. Xpath veri bağlama ifadelerini tanımlayan şablonlar kullanılarak bu işlem yapılabilir.

XmlDataSource3.aspx

```
<form runat="server" action="" method="post">
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataSourceID="XmlDataSource1">
<Columns>
<asp:TemplateField HeaderText="Kitap">
<ItemTemplate>
<b><%# XPath("ad") %></b><br />
<%# XPath("yazar") %><br />
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml" />
</form>
```

Resim 1.19: XmlDataSource3.aspx sayfasının tarayıcıdaki görüntüsü

XmlDataSource kontrolünün bildirimi sırasında kontrolün Xpath özelliği (XmlDataSource.XPath) kullanılarak filtreleme işlemi yapılabilir. Örneğin, sadece yazarları görüntülemek için XmlDataSource kontrolü bildirimi sırasında gerekli Xpath ifadesi kullanılır. Xpath ifadesi Xml belgesinden bir XmlNodeList çıkarır.

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml" XPath="/kitaplar/kitap/yazarlar/yazar" />
```

Xpath ifadesi Xml belgesinden bir XmlNodeList çıkarır. Eğer ifade düğüm listesi geri döndürüyorsa ve bütün veri özellikler içinde bulunuyorsa fazladan işlem yapmaya gerek yoktur. Fakat veri eleman metni içindeyse şablon oluşturulmalıdır. Aşağıdaki örnek her <yazar> düğümünün metnini gösteren şablonu içermektedir.

XmlDataSource4.aspx

```
<form runat="server" action="" method="post">
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataSourceID="XmlDataSource1">
<Columns>
<asp:TemplateField HeaderText="Yazarlar">
<ItemTemplate>
<%# XPath(".") %><br />
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml"
XPath="/kitaplar/kitap/yazarlar/yazar" />
</form>
```

Resim 1.20: XmlDataSource4.aspx sayfasının tarayıcıdaki görüntüsü

FormView_Gosterme.aspx

```
<form id="Form1" runat="server">
<asp:FormView ID="Formview1" runat="server"
  DataSourceID="XmlDataSource1"
  AllowPaging="True" BorderWidth="1px"
  BorderColor="#CC9966"
  BorderStyle="None" BackColor="Blue">
  <RowStyle ForeColor="#3333ff"
  BackColor="#cccccc"></RowStyle>
  <PagerStyle ForeColor="#cc3300" HorizontalAlign="Center"
  BackColor="#FFFFCC"></PagerStyle>
  <ItemTemplate>
    Sıra Nu   :<b>
    <asp:Label ID="lblnu" Runat="server"
    Text='<%# XPath("nu") %>' /></b><br/>
    Kitap Adı :<b>
    <asp:Label ID="lblad" Runat="server"
    Text='<%# XPath("ad") %>' /></b><br />
    Yazar Adı :<b>
    <asp:Label ID="lblyazar" Runat="server"
```

```
        Text='<%# XPath("yazar") %>' /></b><br />
        Yayınevi  :<b>
        <asp:Label ID="lblyayinevi" Runat="server"
        Text='<%# XPath("yayinevi") %>' /></b><br />
      </ItemTemplate>
    </asp:FormView>
    <asp:XmlDataSource id="XmlDataSource1" runat="server"
       DataFile="data/kitaplar.xml" />
    </form>
```

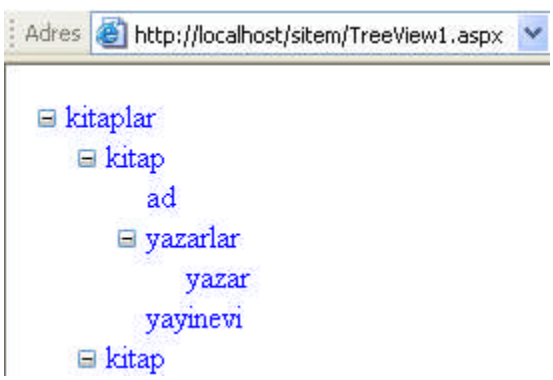
Resim 1.21: Ekran çıktısı

1.14.2. TreeView ile Hiyerarşik Bağlama

TreeView kontrolü ile hiyerarşik bağlama gerçekleştirilir.

TreeView1.aspx

```
    <form runat="server" action="" method="post">
    <asp:TreeView           ID="TreeView1"           runat="server"
DataSourceID="XmlDataSource1" />
    <asp:XmlDataSource ID="XmlDataSource1" runat="server"
    DataFile="data\kitabevi.xml" />
    </form>
```

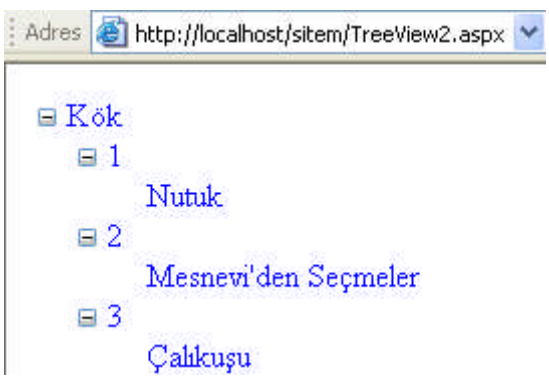


Resim 1.22: TreeView1.aspx sayfasının tarayıcıdaki görüntüsü

TreeView'ın varsayılan XML gösterimi sadece belge yapısını (eleman isimlerini) gösterir. Belgenin içeriğini (eleman metnini) ve özellikleri göstermez. Bunları görüntülemek için TreeView2.aspx sayfasında olduğu gibi TreeView'in **AutoGenerateDataBindings** özelliğini False yapmak ve XML belgesinin istenilen kısımlarına erişmek gerekir.

TreeView2.aspx

```
<form runat="server" action="" method="post">
<asp:TreeView ID="TreeView1" runat="server"
DataSourceID="XmlDataSource1"
AutoGenerateDataBindings="False">
<DataBindings>
<asp:TreeNodeBinding DataMember="kitaplar" Text="Kök" />
<asp:TreeNodeBinding DataMember="kitap" TextField="nu" />
<asp:TreeNodeBinding DataMember="ad" TextField="#InnerText" />
</DataBindings>
</asp:TreeView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="data\kitabevi.xml" />
</form>
```



Resim 1.23: TreeView2.aspx sayfasının tarayıcıdaki görüntüsü

TreeView kontrolünün **<DataBinding>** bölümünde **<TreeNodeDataBinding>** ile hangi verilerin görüntüleneceği belirtilmiştir. Kök elemanı ile başlanmış, ardından istenilen diğer veriler bağlanmıştır.

<TreeNodeDataBinding> elemanının **DataMember** özelliği ile XML belgesindeki bağlanılacak düğüm belirlenmiştir. **TextField** ve **Value** özelliklere bağlanmak için kullanılır. **TextField="nu"** ifadesi ile <nu> özelliğinin metni görüntülenmiştir. **TextField="#InnerText"** ifadesi ile <ad> elemanın içeriği görüntülenmiştir.

FormView_Görüntüleme.aspx

```
<form id="Form1" runat="server">
<asp:FormView ID="Formview1" runat="server"
  DataSourceID="XmlDataSource1"
```

```
  AllowPaging="True" BorderWidth="1px"
  BorderColor="#CC9966"
  BorderStyle="None" BackColor="Blue">
  <RowStyle ForeColor="#3333ff"
  BackColor="#cccccc"></RowStyle>
  <PagerStyle ForeColor="#cc3300" HorizontalAlign="Center"
  BackColor="#FFFFCC"></PagerStyle>
  <ItemTemplate>
    Sıra Nu   :<b>
    <asp:Label ID="lblnu" Runat="server"
    Text='<%# XPath("nu") %>' /></b><br/>
    Kitap Adı :<b>
    <asp:Label ID="lblad" Runat="server"
    Text='<%# XPath("ad") %>' /></b><br />
    Yazar Adı :<b>
    <asp:Label ID="lblyazar" Runat="server"
    Text='<%# XPath("yazar") %>' /></b><br />
    Yayınevi  :<b>
    <asp:Label ID="lblyayinevi" Runat="server"
    Text='<%# XPath("yayinevi") %>' /></b><br />
  </ItemTemplate>
</asp:FormView>
<asp:XmlDataSource id="XmlDataSource1" runat="server"
   DataFile="data/kitaplar.xml" />
</form>
```

Resim 1.24: FormView_Gosterme.aspx sayfasının tarayıcıdaki görüntüsü

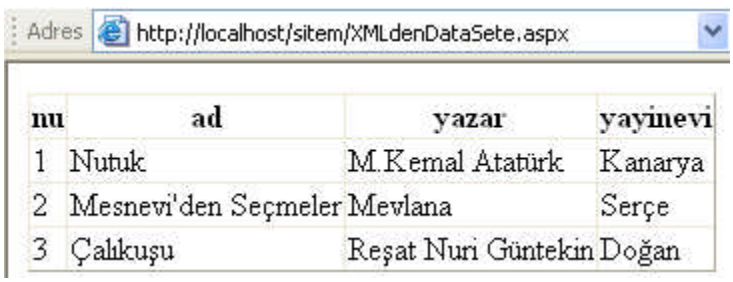
1.15. XML'den DataSet'e Veri Çekme

ADO.NET DataSet'in içeriği XML katarından **(stream)** veya XML belgelerinden elde edilebilir. XML'den verilerle DataSet'i doldurmak için DataSet nesnesinin ReadXML metodu kullanılır. ReadXML metodu XML katarının veya belgesinin içeriğini okur ve DataSet'i verilerle doldurur.

XMLdenDataSete.aspx

```
<%@ Import Namespace="System.Data" %>

<script runat="server">
```

```
    Sub Page_Load(Src As Object, E As EventArgs)
      Dim DataSet1 As New DataSet
      DataSet1.ReadXml(Request.PhysicalApplicationPath                +
"sitem\data\kitaplar.xml")
      GridView1.DataSource = DataSet1
      GridView1.DataBind()
    End Sub
    </script></head>
    <body>
    <form runat="server" name="form1" method="post" action="">
    <asp:GridView ID="GridView1" Runat="server" >
    </asp:GridView>
    </form>
```



nu	ad	yazar	yayinevi
1	Nutuk	M.Kemal Atatürk	Kanarya
2	Mesnevi'den Seçmeler	Mevlana	Serçe
3	Çalıkuşu	Reşat Nuri Güntekin	Doğan

Resim 1.25: XMLdenDataSete.aspx sayfasının tarayıcıdaki görüntüsü

UYGULAMA FAALİYETİ

Aşağıdaki verileri içerecek şekilde XML belgesini oluşturunuz. (Belgenin adı sarf_malzeme.xml olabilir.)

Ürün adı : Yazılabilir DVD-R (5 ADET)
Ürün kodu : 1
Ürün fiyatı : 5 YTL
Stok Durumu : Stokta Mevcut

Ürün adı : Yazılabilir CD-R (100 ADET)
Ürün kodu : 2
Ürün fiyatı : 22 YTL
Stok Durumu : Stokta Yok

Ürün adı : CD Zarfı (100 Adet)
Ürün kodu : 3
Ürün fiyatı : 3 YTL
Stok Durumu : Stokta Yok

Ürün adı : CD/DVD Kutusu (10'lu)
Ürün kodu : 4
Ürün fiyatı : 2 YTL
Stok Durumu : Stokta Mevcut

İşlem Basamakları	Öneriler
Belgenin XML bildirimini oluşturunuz.	Sürüm (version) bilgisi ve kullanılacak dil kodlamasını belirtiniz.
Belgenin ne tür veriler içerdiğini belirten bir açıklama satırı ekleyebilirsiniz.	
Kök elemanı tanımlayınız.	Kök elemanı <urunler> olarak belirleyebilirsiniz.
Yavru elemanı tanımlayınız.	Yavru elemanı <urun> olarak belirleyebilirsiniz.
Alt yavru elemanları tanımlayınız.	Yavru elemanlar <urunAd>, <urunKod>, <fiyat>, <stokDurum> olabilir.
Tüm elemanları kapatınız.	

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki ifadeleri okuyunuz ve doğru olan ifadelerin önündeki boşluğa “D”, yanlış olan ifadelerin önündeki boşluğa “Y” harfi koyunuz.

1) XML ile düz metin dosyaları veri paylaşmak için kullanılabilir. (D/Y)

2) XML, X-Markup Language kelimelerinin kısaltmasıdır. (D/Y)

3) XML dili artık ihtiyacı göremeyen HTML dilinin yerine kullanılmak için yazılmıştır. (D/Y)

4) XSL, eXtensible Stylesheet Language kelimelerinin kısaltmasıdır. (D/Y)

5) XML verilerini tanımlamak için XML Schema kullanır. (D/Y)

6) Aşağıdakilerden hangisi XML sürümünü tanımlayan bildirimdir?
   A) <xml version="1.0" />
   B) <?xml version="1.0"?>
   C) <?xml version="1.0" />
   D) <# xml version="1.0" #>

7) Aşağıdakilerden hangisi doğrudur?
   A) XML elemanları küçük harfle yazılmalıdır.
   B) XML elemanları kapatılmalıdır.
   C) XML’de etiketler büyük küçük harf duyarlı değildir.
   D) XML’den yeni diller üretilemez.

8) Aşağıdakilerden hangisi doğrudur?
   A) XML belgeleri sadece Not Defteri programıyla oluşturulabilir.
   B) XML verilerini tanımlamak için XSL kullanılır.
   C) XML elemanları alt elemanlara sahip olamazlar.
   D) XML belgeleri kök elemana sahip olmalıdır.

9) Aşağıdakilerden hangisi doğru bir XML etiketidir?
   A) <123> B) <Not>
   C) <1daire> D) <program ismi>

10) Aşağıdakilerden hangisi doğru bir XML etiketidir?
   A) <baba adı>
   B) <MESLEK>
   C) <öğrenim_durumu>
   D) <ad soyad>

11) 11)Aşağıdakilerden hangisi yanlıştır?
   A) XML, HTML’nin bir üst sürümüdür.
   B) XML, verileri taşımak için tasarlanmıştır.
   C) HTML, verileri göstermekle ilgilenir.
   D) XML, verileri tanımlamakla ilgilenir.

ÖĞRENME FAALİYETİ–2

AMAÇ

Web servislerini kullanabileceksiniz.

ARAŞTIRMA

- İnternet üzerindeki web servislerinden örnekler bulunuz.

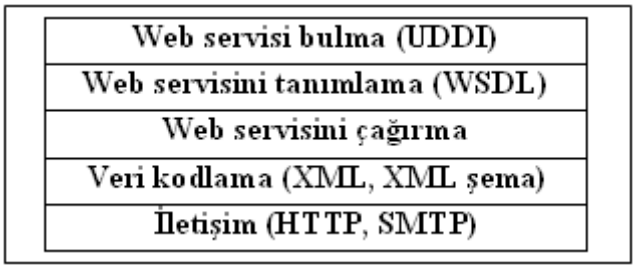
2. WEB SERVİSLERİ

2.1. Web Servisleri

Web servisi, bilgisayarlar arasında ağ üzerinden etkileşimi ve uyumluluğu sağlayacak yazılım sistemidir. Yerel ağ ve internet üzerinden çağrılabilecek yazılım bileşenlerdir. Web servisiyle etkileşim kurmak için HTTP üzerinden XML mesajı gönderilir. Web servisi ile, başkası tarafından geliştirilmiş bir yazılım kullanılabilir, böylelikle bu yazılımın tekrar yazılması gerekmez. Bu teknik programcıların başkası tarafından oluşturulmuş sınıf, bileşen, API kütüphanelerini kullanmasına benzer. Temel fark, web servisinin uzaktaki farklı bir sunucuda yer almasıdır. Bir web servisi internet üzerinden uzaktan çağrılabilir.

Web servisi, XML tabanlı mesajlaşmayı esas aldığından haberleşecek sistemlerin birbiriyle uyumlu olması gerekmez. Örneğin, Java ile geliştirilmiş ve UNIX sistem üzerinde çalışan bir uygulama ile .NET ile geliştirilmiş ve Windows işletim sistemi üzerinde çalışan bir uygulama, birbirlerinin çalışma ortamlarından bağımsız olarak, XML iletişim standartları aracılığıyla iletişim kurabilir. Yani, farklı şirketlere ait veya farklı platformlar üzerinde çalışan uygulamalar arasında fonksiyonellik paylaşılabilir. .NET kullanılarak oluşturulmuş bir web servisi Java istemci uygulaması tarafından çağrılabilir veya bir Java web servisi .NET uygulamasından çağrılabilir.

İnternet üzerinde farklı web servisleri kullanılmaktadır. Örneğin, bir e-ticaret sitesinin nakliye ücretini hesaplamak için bir nakliye firmasının web servisini kullanması, bir haber sitesinin farklı bir haber sitesinin haber başlıklarını ve makaleleri kendi sitesinde yayınlaması,



Şekil 2.1: Web servisi teknolojisi

Web servisleri oluşturma ve kullanma açısından bilinmesi yararlı web servisi standartları aşağıda verilmiştir.

Standart	**Tanımı**
WSDL	Bir web servisi için arayüz tanımlama oluşturmak amacıyla kullanılır. WSDL belgesi, web servisinde hangi metodların var olduğunu, her metodun kullandığı parametre ve geri dönen değerleri, onlarla nasıl iletişim kurulacağını istemciye söyler.
SOAP	Bir web servisiyle iletişim kurulduğunda veri değerleri gibi bilgileri kodlamak için kullanılan mesaj formatıdır.
HTTP	Tüm web servisi iletişiminin üzerinde gerçekleştiği protokoldur. Örneğin, SOAP mesajları HTTP kanalları üzerinden gönderilir.
UDDI	Şirketleri, şirketlerin sunduğu web servislerini, onların WSDL sözleşmeleri için ilişkili URL'leri kataloglayan kayıt servisi oluşturmak için kullanılan standarttır.

Tablo 1.1: Web servisi standartları

2.2. Web Servislerini Bulma

Kullanılmak istenilen web servisinin URL'si biliniyorsa gerekli kodlar yazılarak bu servis kullanılabilir. Kimi durumlarda ise web servisini aramak gerekir. UDDI (Universal Description, Discovery and Integration) kayıt servisi sayesinde kurumlar ihtiyaç duydukları servisleri arayabilmekte veya kendi servislerini farklı kurumlar tarafından bulunabilir hale getirebilmektedir.

UDDI, şirketler tarafından yayınlanan web servislerinin nerede olduğunu gösteren merkezileştirilmiş dizindir. Farklı organizasyonlar ve şirket grupları farklı UDDI kayıt servisi kullanabilir. UDDI dizininden bilgi almak veya bir bileşen kaydetmek (register) için web servisi arayüzü kullanılır.

Özetle, dünya üzerinde birçok web servisi vardır (google, live, amazon, weather vs.). Dünya üzerinde hangi web servisleri vardır ve bunların WSDL dökümanları nerededir gibi bilgilerin depolandığı sunucuya UDDI sunucusu denir.

2.3. Web Servisini Tanımlama

Bir web servisine nasıl erişileceğini bilmek isteyen istemci için servisin hangi metodları içerdiği, her metodun hangi parametreleri kullandığı, her parametrenin veri tipi bilinmelidir. WSDL (Web Service Description Language), tüm bu ayrıntıları tanımlayan XML tabanlı dildir. WSDL (Web Servisi Tanımlama Dili), istemcinin web servisine göndermesi (submit) gereken istek mesajını ve web servisinin geri döndüreceği cevap mesajını tanımlar. WSDL, ayrıca web servisinin yeri ve kullanılması gereken protokolü (genellikle HTTP) tanımlar. Özetle web servisleri, WSDL ile sundukları servisin tanımını yaparlar.

2.4. Mesaj Biçimi

Bir web servisiyle iletişim kurmak için herhangi bir platform üzerinde ayrıştırılabilen ve anlaşılabilen istek ve cevap mesajların oluşturulmasına ihtiyaç vardır. Bu mesajları oluşturmak için kullanılan SOAP, XML tabanlıdır. SOAP, veri alış verişinde kullanılan mesajı (mesaj formatını) tanımlar, fakat mesajın nasıl gönderileceğini tanımlamaz. ASP.NET web servislerinde taşıma protokolü HTTP'dir. Diğer bir deyişle, bir web servisiyle iletişim kurmak için istemci HTTP bağlantısını açar ve SOAP mesajı gönderir.

Bir web servisinin yaşam döngüsü şu şekildedir : İlk olarak, web servisi kullanıcısı web servisinin URL'sine doğrudan giderek veya UDDI sunucusunu kullanarak veya DISCO dosyasını kullanarak web servisini bulur. Sonra istemci, web servisiyle nasıl iletişim kurulacağını tanımlayan web servisinin WSDL belgesini alır. Bu işlemler her ikisi de tasarım zamanında gerçekleştirilir. Uygulama çalıştırılıp web servisiyle iletişime geçilince istemci, uygun web metodunu başlatmak için SOAP mesajını gönderir.

Resim 2.1: ASP.NET uygulaması

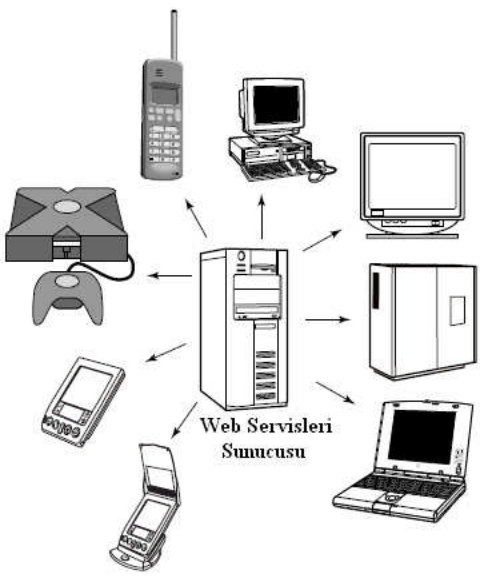
2.5. Web Servisi Çalışma Sistemi

Bu başlığa kadar web servisinin ne olduğu, hangi teknolojilerden oluştuğu anlatılmıştır. Şimdi bir bütün olarak web servisinin nasıl çalıştığını görelim.

Hazırladığınız sitenize arama bölümü koymak istiyorsunuz diyelim. Arama işlemini gerçekleştiren bir program yazmak yerine bir arama sitesinin (örneğin, www.google.com) web servisini kullanmak istiyorsunuz.

Sadece sizin siteniz değil internette birçok site arama sitesinden bu hizmeti alabilir. Arama sitesi, sadece arama servisini (hizmetini) servisten yararlanmak isteyen tüm sitelere sunacaktır fakat sahip olduğu diğer veri tabanı bilgilerine veya programlara bu sitelerin erişimine izin vermeyecektir.

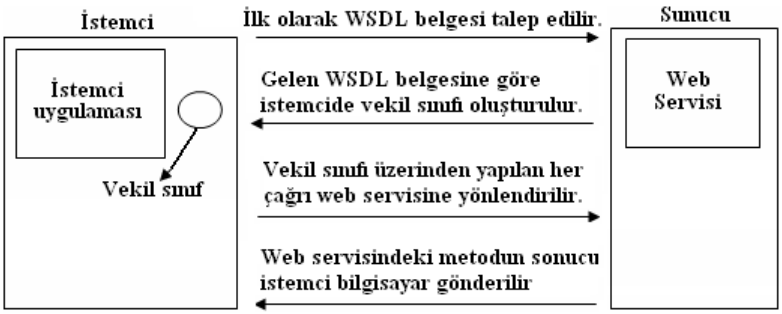
Dikkat edilirse arama sitesinin servisini farklı siteler almaktadır. Arama sitesi ve diğer siteler farklı dillerde oluşturulmuş ve farklı platformlar üzerinde çalıştırılıyor olabilirler. Web servisini bir Java uygulaması da, Windows uygulaması da, bir cep bilgisayarı da kullanabilir.



Resim 2.2: Web servislerinin farklı istemciler tarafından kullanılması

Web servisi genel olarak bir sınıftan oluşur. Sınıf içerisine konulan metod servisin yapacağı işlemi gerçekleştirir. Servisten yararlanmak isteyenlerle bu metot web'den yayınlanarak paylaşılır. Bu aşamada kullanılan programlama dili farkından kaynaklanan bir uyumsuzluk sorunu vardır. Örneğin, arama sitesindeki sınıfın C# dilinde yazıldığını ve istemci uygulamasının ise java dilinde yazılmış olduğunu kabul edelim. Bu durumda uyumsuzluk sorun ortaya çıkacaktır. Java dilinde C# sınıfının nesnesi oluşturulup kullanılamaz.

Bu problemi aşmak için şöyle bir yol izlenmektedir. Web servisi sınıfının tanımını içeren bir dosya oluşturulur. Bu dosya, sınıfın ismini, metodlarını, metodun aldığı parametreleri, türleri, geri döndürdüğü değerler, türleri gibi sınıfın bir tanımını içerir. Bu dosya hem Java'dan hem C#'dan hem de Visual Basic dilinden anlaşılabilecek bir dille yazılmış olması gerekir. Bu dilin adı **WSDL**'dir. Böylelikle web servisi sınıfının tanımını içeren wsdl belgesini hangi istemci alırsa sınıfla ilgili her şeyi bilecektir.



Şekil 2.2: Web servisini kullanma

İstemci, sunucudan aldığı WSDL belgesine göre C# diliyle geliştirilmiş sınıfın benzerini Java diliyle oluşturacaktır. Peki istemcideki Java ile sunucudaki C# kendi aralarında nasıl anlaşacaklar?

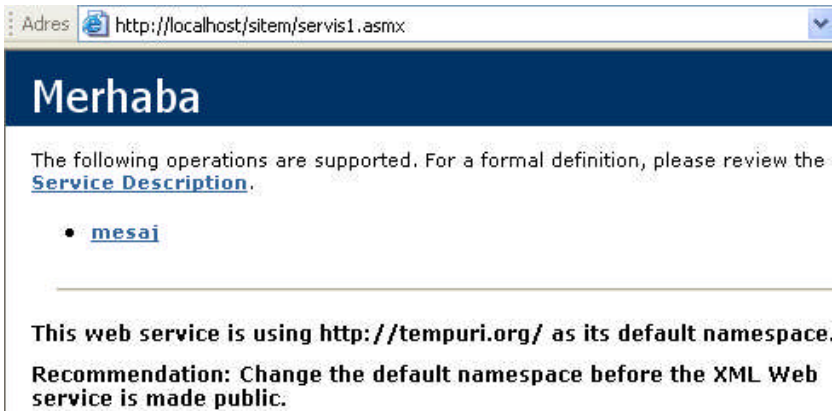
İki taraf arasındaki anlaşmayı sağlayan SOAP'dır. Java'da yazılmış sınıfın ilgili metodu çağrıldığında bu istek SOAP paketine dönüştürülüp sunucuya gönderilecektir. Sunucu, SOAP paketini alıp inceledikten sonra istemcinin ilgili metodu çağırdığını anlayacaktır. Ardından C# diliyle yazılmış sınıfının bir nesnesi oluşturulup metodu işletilecektir. Metodun işletilmesiyle elde edilen veri istemciye SOAP paketleriyle gönderilecektir.İstemci SOAP paketi aldıktan sonra içindeki veriyi Java'da işleyecektir.

İstemci uygulamada, verilen talepleri SOAP paketine çevirip web servisinin ilgili metodu işletmesi yönünde komut veren, aracılık eden bu sınıfa **proxy class** (vekil sınıf) denilmektedir.Yani WSDL'e göre oluşturulan sınıf vekil sınıf olarak isimlendirilmektedir. Vekil sınıf kimi kaynaklarda temsilci sınıf, aracı sınıf olarak geçmektedir.

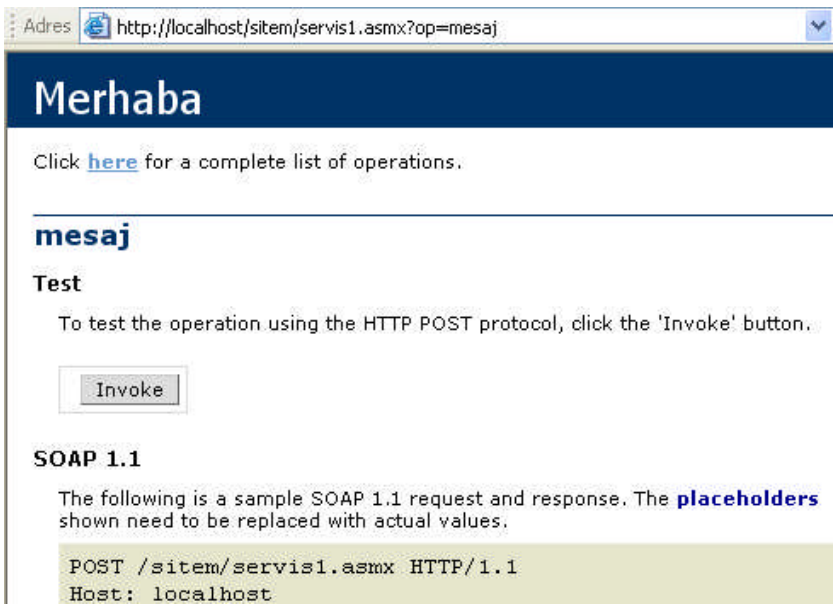
2.6. Basit Bir Web Servisi Oluşturma

Tarayıcıya "Herkese Merhaba" yazdıran basit bir web servisi oluşturalım. Tamamen boş olan bir sayfaya aşağıdaki kodları yazınız. Ardından **servis1.asmx** adıyla kaydedip tarayıcıda görüntüleyiniz.

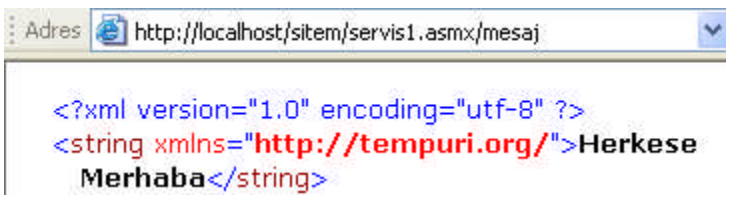
```
<%@ WebService Language="VB" Class="Merhaba" %>
Imports System.Web.Services
Public Class Merhaba:inherits WebService
<WebMethod()> _
Public Function mesaj As String
Return("Herkese Merhaba")
End Function
End Class
```



Resim 2.3: servis1.asmx dosyasının tarayıcıdaki görüntüsü



Resim 2.4: Mesaj linkine tıklandığında



Resim 2.5: Invoke düğmesine tıklandığında

UYGULAMA FAALİYETİ

İki sayının toplamını veren bir web servisi oluşturunuz.

İşlem Basamakları	Öneriler
➢ Web servis bildirimini yazınız.	➢ <%@WebService...
➢ System.Web.Services ad alanını sayfaya dahil ediniz.	➢ Imports… ➢
➢ Sınıfı tanımlayınız.	➢ Public Class...
➢ Metodu tanımlayınız.	➢ <WebMethod(…
➢ Fonksiyonu tanımlayınız.	➢ Function…
➢ Fonksiyonun sonucu geri gönderiniz.	➢ Return…
➢ Fonksiyon ve sınıfı sonlandırınız.	➢ End Function….

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki ifadeleri okuyunuz ve doğru olan ifadelerin önündeki boşluğa “D”, yanlış olan ifadelerin önündeki boşluğa “Y” harfi koyunuz.

1) Web servislerinin uzantısı aspx’dir.(D/Y)

2) İhtiyaç duyduğumuz bir web servisini bulmak için SOAP kullanılır. (D/Y)

3) Sunucudan alınan WSDL belgesine göre vekil sınıfı oluşturulur.(D/Y)

4) Web servisleri istemci üzerinde çalışan program parçalarıdır. (D/Y)

5) SOAP, HTML tabanlı bir dildir.(D/Y)

MODÜL DEĞERLENDİRME

Aşağıdaki ifadeleri okuyunuz ve doğru olan ifadelerin önündeki boşluğa “D”, yanlış olan ifadelerin önündeki boşluğa “Y” harfi koyunuz.

1) XML, verileri metin tabanlı tanımlar. (D/Y)

2) XML’de veri ile verinin sunumu birbirinden ayrıdır. (D/Y)

3) XML verilerini biçimlendirmek için sadece CSS biçim dosyaları kullanılır. (D/Y)

4) XML yeni diller oluşturmak için kullanılamaz. (D/Y)

5) XML’de bazı karakterler özel anlam taşır. (D/Y)

6) XML söz dizimi kurallarına uyan XML belgelerine iyi biçimli XML belgesi denir. (D/Y)

7) Uygulamaların XML belgelerini işlemesi için kullanılan yazılımlara XML DOM denir. (D/Y)

8) Web servisi, yerel ağ ve internet üzerinden çağrılabilecek yazılım bileşenlerdir. (D/Y)

9) UDDI, bir web servisi için arayüz tanımlama oluşturmak amacıyla kullanılır. (D/Y)

10) 10) WSDL, veri alış verişinde kullanılan mesajı (mesaj formatını) tanımlar. (D/Y)

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği aşağıdaki ölçütlere göre değerlendiriniz

Değerlendirme Ölçütleri	Evet	Hayır
1. HTML ile XML arasındaki farkları açıklayabilir misin?		
2. Kurallara uygun XML belgesi oluşturabilir misin?		
3. Gerektiğinde XML ad alanlarını kullanabilir misin?		
4. XML belgesinin yapısını tanımlamak için XML Schema belgesini oluşturabilir misin?		
5. XML belgesini biçimlendirmek için XSL belgesini oluşturabilir misin?		
6. XML belgesinde dolaşmak için XPATH ifadelerini kullanabilir misin?		
7. Kod kullanarak XML belgesi yazabilir misin?		
8. Kod kullanarak XML belgesi okuyabilir misin?		
9. Kod kullanarak XML belgesini görüntüleyebilir misin?		
10. XML veri bağlama işlemini gerçekleştirebilir misin?		
11. XML belgesinden DataSet'e veri çekebilir misin?		
12. Bir web servisi oluşturabilir misin?		

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ – 1'İN CEVAP ANAHTARI

Sorular	Cevaplar
1	D
2	Y
3	Y
4	D
5	D
6	B
7	B
8	D
9	B
10	C
11	A

ÖĞRENME FAALİYETİ – 2'NİN CEVAP ANAHTARI

Sorular	Cevaplar
1	Y
2	Y
3	D
4	Y
5	Y

MODÜL DEĞERLENDİRME CEVAP ANAHTARI

Sorular	Cevaplar
1	D
2	D
3	Y
4	Y
5	D
6	D
7	Y
8	D
9	Y
10	Y

SÖZLÜK

Terim	Okunuşu	Anlamı, Açıklaması
Cross-platform	Kros pletform	Çok platformlu. Bir yazılım ya da donanımın farklı tür makineler üzerinde çalışabilmesi; örneğin bir kodun gerek Macintosh gerekse IBM uyumlu elemansel bilgisayarlarda yürütülebilmesi.
Element	Elımınt	Öge, eleman, HTML, SGML, XML gibi dillerde bir etiket kümesi, bu etiketlerin özellikleri ve etiketler arasında kalan bilgi içeriği.
incompatible	İnkımpedıbıl	Bağdaşmayan, uyumsuz. Aralarında uyum olmayan yazılım ya da donanıma ilişkin
parse	pars	Ayrıştırmak. 1) Bir algoritma ya da bilgisayar programı aracılığı ile tümceyi dilin biçimbilgisi, sözdizimi, gerekirse anlambilgisi kurallarına göre, basamak basamak işlevsel öğelerine ayırmak. 2) Bir bilgisayar programının veriler üzerinde işlem yapmasını sağlayacak şekilde girdiyi küçük parçalara ayırma. Örneğin, blokları deyimlere, deyimleri ifadelere, ifadeleri de işleç ve işlenenlere ayırma.
Read	Rid	Okumak
Schema	Sikima	Şema. Bir veri tabanının mantıksal yapısının betimlemesi.
Syntax	Sinteks	Söz dizimi, sentaks. Bir dilin, özellikle bir programlama dilinin deyim ve tümcelerini oluşturan karakter dizgilerinin nasıl üretileceğini ve ilişkilendirileceğini tanımlayan kurallar kümesi.
Universal resource identifier	Yunivörsıl Risors aydentifayır	Evrensel kaynak tanılayıcısı. Internet nesnelerini tanılamaya ve erişmeye yarayan türlerine özgü ad ve adresler kümesi.En yaygın evrensel kaynak tanılayıcıları URL ve bağıl URL türleridir.

KAYNAKÇA

- DEMİRKOL Zafer, "**ASP.NET"**, Pusula Yayıncılık, İstanbul, 2005
- DUTHIE G. Andrew, "**Adım Adım Microsoft"** ASP.NET, Arkadaş Yayınevi, Ankara, 2005
- SANKUR Bülent, "**Bilişim Sözlüğü 2005 Programı"**, Yazılım: Hakan GÜLERYÜZ, Pusula Yayıncılık
- PALA Zeydin, "**ASP.NET İle Adım Adım"** Web Uygulamaları, Türkmen Kitabevi, İstanbul, 2006
- ÖZBA**Y** Sabahat, AKYAZI Selma, "**Elektronik Ticaret"**, Sistem Ofset Yayıncılık, Ankara.
- KABAKÇI Şenel, TAŞKIN Gaffar**, "Elektronik Ticaret"**, Yeni Çizgi Yayınları. Ankara.
- Introducing .NET and ASP.NET. Bill Evjen.
- Pro ASP.NET 2.0 in VB 2005. Laurence Moroney, Matthew MacDonald. Apress
- XML Integration with Relational Data and ADO.NETmsdn2.microsoft.com/en-us/library/aa735761(VS.71).aspx.
- Loading a DataSet from XML. msdn2.microsoft.com/en-us/library/aa720235(VS.71).aspx
- Synchronizing a DataSet with an XmlDataDocument.

msdn2.microsoft.com/en-us/library/aa720681(VS.71).aspx

- XML Data Sources.

msconline.maconstate.edu/tutorials/aspnet20/ASPNET03/aspnet03-04.aspx.