BLM3580
Sistem Programlama

2021-2022 GÜZ YARIYILI

DR.ÖĞR.ÜYESI GÖKSEL BIRICIK

XML-XSD

XML Schema (XSD)

XML Schema, XML belgelerinin doğrulanmasında DTD'ye bir alternatiftir. Aynı zamanda XML Schema Definition (XSD) olarak da bilinir.

XML belgesi içinde yer alacak elemanları tanımlar

Elemanların içinde yer alacak olan özellikleri tanımlar

Belge içinde hangi elemanların alt-elemanlar (child) olduğunu belirler

Alt-elemanların sıralarını belirler

Elemanların ne tip bir metin içereceklerine veya boş olup olmadıklarını belirler

Elemanlar ve özniteliklerinin veri tiplerini belirler.

Elemanlar ve özniteliklerinin önceden belirlenmiş (default) ve sabit (fixed) değerlerini belirler

XML Schema (XSD)

XML Schema, Microsoft tarafından geliştirilmeye başlanmış ancak 2001 yılında W3C tarafından resmileştirilmiştir.

Veri yapılarını destekler, DTD'nin sağlayamadığı özellikleri sunar, DTD'den daha kullanışlıdır, ileride yapılabilecek eklemelere açıktır.

Veri yapılarını destekler

- Verinin doğruluğunun kontrol edilmesi kolaylaşmıştır.
- Kısıtlamalar tanımlanarak (restrictions) veri üzerindeki kontrol arttırılır.
- Verinin ne şekilde verilebileceğini belirleyen formatlar (patterns) tanımlanabilir.
- Veriyi farklı veri tiplerine dönüştürme imkânı sağlar

XML ile yazılır

- Bu sayede yeni bir kodlama dil öğrenmeye gerek kalmaz
- Mevcut XML editörlerinin kullanılmasına devam edilebilir
- Mevcut XML ayrıştırıcılarının kullanılmasına devam edilebilir.
- Schema üzerinde XML-DOM ile işlemler yapmak mümkündür.
- XSLT yardımıyla schema üzerinde dönüşümler (transformation) yapmak mümkündür.

Namespace destekler.

<schema> etiketi

XML belgeleri tek bir kök elemana bağlıdır.

Schema da bir XML belgesidir.

- Kök elemanı <schema> dır

<?xml version = “1.0” encoding = “UTF-8”?>

<xs:schema xmlns:xs = “http://www.w3c.org/2001/XMLSchema”>

…

</xs:schema>

XSD ile XML Doğrulama

XSD, XML içinde dahili olarak kullanılamaz. Namespace içinde doğrulama için kullanılacak XSD dosyası belirtilmelidir.

```
<?xml version = “1.0” encoding = “UTF-8” ?>
<note    xmlns      = “http://www.ce.yildiz.edu.tr”
         xmlns:xsi = “http://www.w3c.org/2001/XMLSchema-instance”
         xsi:schemaLocation= “http://www.ce.yildiz.edu.tr/note.xsd”>
                  <to>öğrenciler</to>
                  <from>öğretmen</from>
                  <subject>Sınav tarihleri</subject>
                  <body>Sistem programlama dersinin sınav tarihleri ilan edilmiştir</body>
</note>
```

XSD basit eleman (simple element)

Tanımlanan elemanın altında başka elemanlar veya öz nitelikler bulunmuyor ise basit elemandır.

Veri Tipi	Açıklama
xs:string	Bu tipte tanımlanmış elemanlar karakter (boşluk sekme, CR, LF de olabilir) içerir
xs:decimal	18 haneye kadar sayısal bir değer ifade etmek üzere kullanılır. (15 25.75 -2.35 99986 gibi)
xs:integer	Kesirli olmayan bir sayı ifade etmek üzere kullanılır.
xs:float	Kesirli sayıları ifade etmek üzere kullanılır (32 bit). 1267.43233E12, 12.78e-2, 12
xs:double	Kesirli sayıları ifade etmek üzere kullanılır (64 bit)
xs:boolean	{true, false} değerlerinden birini ifade etmek üzere kullanılır.
xs:date	YYYY-MM-DD şeklinde ifade edilir.
xs:time	hh:mm:ss şeklinde ifade edilir.
xs:dateTime	YYYY-MM-DDThh:mm:ss şeklinde ifade edilir.
xs:duration	Aradaki zaman farkını ifade etmek için kullanılır. P1Y2M3DT10H30M20S veya –P120D gibi

XSD basit eleman (simple element)

<xs:element name = “*eleman_ismi*” type = “*veri_tipi*” />

<xs:element name = “*ad*” type = “xs:string” /> <ad>Göksel</ad>

<xs:element name = “*yaş*” type = “xs:integer” /> <yaş>40</yaş>

<xs:element name = “*sınavtarihi*” type = “xs:date” /> <sınavtarihi>2021-11-24</sınavtarihi>

Önceden belirlenmiş “default” (aksi belirtilmedikçe varsayılan değer) ve “fixed” (elemanın mutlaka taşıması gereken değer) değer tanımlanabilir.

<xs:element name = “*kur*” type = “xs:string” **default** = “*TL*” />

<xs:element name = “*kur*” type = “xs:string” **fixed**= “*TL*” />

Schema Özellikleri

<xs:attribute name = “*attribute_ismi*” type = “*veri_tipi*” />

Her zaman basit (simple) tipte tanımlamalar ile ifade edilirler.

Bir elemanın özniteliği var ise, karmaşık (complex) elemandır. (Tanımını göreceğiz)

<xs:attribute name = “*hitap*” type = “xs:string” /> → <alıcı hitap=“Bay”>Göksel Biricik</alıcı>

Elemanlar gibi “default” ve “fixed” öntanımlı değerleri de olabilir.

Özellikler seçimliktir. Bu durumu açıkça belirtmek ya da zorunlu olduğunu belirtmek için:

<xs:attribute name = “*id*” type = “xs:decimal” use = “*optional*” /> →<ad>Göksel</ad>

<xs:attribute name = “*id*” type = “xs:decimal” use = “*required*” /> →<ad id=“007”>James</ad>

Schema sınırlamaları (restrictions/facets)

DTD ile sağlayamayacağımız kısıtları tanımlayabiliriz. Ör. Yaş:0-100

```
<xs:element name = “yaş”>
        <xs:simpleType>
                <xs:restriction base = “xs:integer”>
                        <xs:minInclusive value = “0” />
                        <xs:maxInclusive value= “100” />
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Schema sınırlamaları (restrictions/facets)

String tipinde enumeration ile kısıt koyabiliriz.

```
<xs:element name = “tatlılar”>
<xs:simpleType>
        <xs:restriction base = “xs:string”>
                <xs:enumaration value = “kadayıf” />
                <xs:enumaration value = “kazandibi” />
                <xs:enumaration value = “baklava” />
                <xs:enumaration value = “dondurma” />
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Schema sınırlamaları (restrictions/facets)

Kod ve uzunluk kısıtlamalarını enumeration ile yapamayız. Pattern ile sağlayabiliriz.

Harf elemanının sadece bir uzunlukta küçük harf olması kısıtı:

```
<xs:element name = "harf">
        <xs:simpleType>
                <xs:restriction base= "xs:string">
                        <xs:pattern value= "[a-z]" />
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Schema sınırlamaları (restrictions/facets)

Kod ve uzunluk kısıtlamalarını enumeration ile yapamayız. Pattern ile sağlayabiliriz.

Kod elemanının 3 büyük harften olması kısıtı:

```
<xs:element name = "kod">
        <xs:simpleType>
                <xs:restriction base= "xs:string">
                        <xs:pattern value= "[A-Z][A-Z][A-Z]" />
                </xs:restriction>
        </xs:simpleType>
</xs:element
```

Schema sınırlamaları (restrictions/facets)

Kod ve uzunluk kısıtlamalarını enumeration ile yapamayız. Pattern ile sağlayabiliriz.

Seçenek elemanının sadece {a,b,c,d,e} elemanlarından biri olabilmesi kısıtı:

```
<xs:element name = “seçenek”>
        <xs:simpleType>
                <xs:restriction base= “xs:string”>
                        <xs:pattern value= “[abcde]” />
                </xs:restriction>
        </xs:simpleType>
</xs:element
```

Schema sınırlamaları (restrictions/facets)

Kod ve uzunluk kısıtlamalarını enumeration ile yapamayız. Pattern ile sağlayabiliriz.

Şifre elemanının 8 karakter ve büyük-küçük harf ve sayıdan oluşabilmesi kısıtı:

```
<xs:element name = "şifre">
        <xs:simpleType>
                <xs:restriction base= "xs:string">
                        <xs:pattern value= "[a-zA-Z0-9]{8}" />
                </xs:restriction>
        </xs:simpleType>
</xs:element
```

Schema ve Whitespace

```
<xs:element name = “adres”>
        <xs:simpleType>
                <xs:restriction base= “xs:string”>
                        <xs:whiteSpaces value= [“preserve“|“replace“|“collapse“] />
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Preserve: *adres* ismindeki eleman içindeki beyaz boşluk (white space) karakterleri XML ayrıştırıcısı tarafından kaldırılmaz.

Replace: Beyaz boşluk (white space) karakterleri XML ayrıştırıcısı tarafından boşluk ile değiştirilir.

Collapse: Beyaz boşluk (white space) karakterleri, boşluk ile değiştirilir, cümle başındaki ve sonundaki boşluklar kaldırılır, birden fazla olan boşluklar teke indirilir.

Schema ile Değer Kısıtlamaları

Tipleri	Açıklama
enumeration	Kabul edilebilecek değerlerin listesini belirler.
fractionDigits	Noktadan sonra kaç hane olacağını belirler Değeri 0 veya daha büyük olmalıdır.
length	Tam olarak kaç karakter olduğunu veya kaç tane liste elemanının seçilebileceğini belirler.
maxExclusive	Sayısal değerler için hariç tutulacak üst sınırı belirler.
maxInclusive	Sayısal değerler için dâhil edilebilecek üst sınırı belirler.
maxLength	Kullanılabilecek azami karakter veya liste elemanı sayısını belirler
minExclusive	Sayısal değerler için hariç tutulacak alt sınırı belirler
minInclusive	Sayısal değerler için dâhil edilebilecek alt sınırı belirler
minLength	Kullanılabilecek asgari karakter veya liste elemanı sayısını belirler
pattern	Kabul edilebilir bilginin formatını belirler
totalDigits	Kabul edilebilecek hane sayısını belirler
whiteSpaces	Boşluk karakterlerinin (CR, LF, tabs, space) nasıl değerlendirileceğini belirler.

Schema Karmaşık Eleman (Complex Element)

Karmaşık elemanlar, kendi altında başka elemanlar ve özellikler bulundurabilir.

- Boş, kendi altında başka elemanlar bulunduran, sadece metin bulunduran ve hem metin hem de kendi altında başka elemanlar bulunduran türleri olabilir.

Schema Complex Element Tanımları	XML Örnekleri / Açıklama
<xs:element name = “hoca” type = “kişiselbilgiler” /> <xs:complexType name = “adsoyad”> <xs:sequence> <xs:element name = “ad” type = “xs:string” /> <xs:element name= “soyad” type = “xs:string” /> </xs:sequence> </xs:complexType> <xs:complexType name = “kişiselbilgiler”> <xs:complexContent> <xs:extension base = “adsoyad”> <xs:sequence> <xs:element name = “adres” type = “xs:string” /> <xs:element name = “email” type = “xs:string” /> </xs:sequence> </xs:extension > </xs:complexContent> </xs:complexType>	<hoca> <ad>Göksel</ad> <soyad>Biricik</soyad> <adres>YTÜ</adres> <email>gbiricik@yildiz.edu.tr</email> </hoca> Örnekte hoca isimli eleman için kişiselbilgiler isimli bir veri tipi tanımlanmıştır. Bu veri tipi adsoyad isimli veri tipini xs:extension ile genişletmiş ve buna adres ile email bilgilerini de eklemiştir. xs:complexContent ile kişiselbilgiler isimli karmaşık yapının adsoyad isimli karmaşık yapı ile birleştirilmesi mümkün olmuştur.

Schema Karmaşık Eleman (Complex Element)

Karmaşık elemanlar, kendi altında özellikler barındırmaları durumunda xs:complexContent yerine xs:simpleContent tanımı kullanılır.

Schema Complex Element Tanımları	XML Ornekleri / Açıklama
<xs:element name="Telefon" type="Telefontype" maxOccurs="3"/> <xs:complexType name="Telefontype"> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Tipi"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="Ev"/> <xs:enumeration value="İş"/> <xs:enumeration value="Cep"/> </xs:restriction> </xs:simpleType> </xs:attribute> </xs:extension> </xs:simpleContent> </xs:complexType>	Tanım uyarınca Telefon isimli eleman en fazla 3 defa tekrarlanabilir. Telefon elemanının özniteliği olan Tipi'nin alabileceği değerler Ev, İş ve Cep olarak belirlenmiştir. <Telefon Tipi="Ev"> 0-216-1234567 </Telefon> <Telefon Tipi="İş"> 0-212-1234567 </Telefon> <Telefon Tipi="Cep"> 0-555-1234567 </Telefon>

Schema Karmaşık Eleman (Complex Element)

Karmaşık elemanların içinde veri olması da istenebilir.

- *complexType* tanımında, eleman içinde başka değerler de olabileceğini belirlemek üzere *mixed= "true"* ibaresi kullanılmalıdır.

```
<xs:element name = "mektup" >
<xs:complexType mixed = "true">
  <xs:sequence>
    <xs:element name = "ad" type = "xs:string"/>
    <xs:element name= "tarih" type = "xs:date"/>
    <xs:element name= "odeme" type = "xs:decimal"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

```
<mektup>
Sayın <ad>Göksel Biricik</ad> yapmış olduğunuz <tarih>2021-12-01</tarih> tarihli ödemenizden sonra <odeme>5000000</odeme> TL borcunuz kaldığını bildiririz.
</mektup>
```

XSD karmaşık elemanların belge içinde kullanımı

XSD ile elemanların hangi sırada (order) yerleştirileceği, kaç defa tekrarlanacağı (occurence) ve nasıl gruplanacağı kontrol edilebilir.

1. Sıralama
   - a.All: alt-elemanların her biri, bir defa tekrarlanmak üzere istenilen sırada yazılabilir.
   - b.Choice: alt-elemanlardan sadece biri yer alabilir.
   - c.Sequence: alt-elemanların belirlenen sırada yer almasını sağlar.

2. Tekrar
   - a.maxOccurs: Bir elemanın en fazla kaç defa tekrarlanabileceğini belirler.
   - b.minOccurs: Bir elemanın en az kaç defa tekrarlanabileceğini belirler.

3. Gruplama
   - a. Group name: Karmaşık yapıda elemanların altında yer alan alt-elemanların yapısını belirlemek amacıyla kullanılır.
   - b. attributeGroup name: Elemanların gruplanmasına benzer bir uygulama öznitelikler için de yapılabilir.

XSD Eleman Sıralama Örnekleri

XSD Tanımlamaları	Açıklama / Örnek
<xs:element name= “kişi”> <xs:complexType> <xs:all> <xs:element name= “ad” type= “xs:string” /> <xs:element name= “soyad” type= “xs:string” /> </xs:all> </xs:complexType> </xs:element>	ad ve soyad isimli elemanların bir kere ve istenen sırada yazılabilirler <kişi> <ad>Göksel</ad> <soyad>Biricik</soyad> </kişi> veya <kişi> <soyad>Biricik</soyad> <ad>Göksel</ad> </kişi>
<xs:element name= “kişi”> <xs:complexType> <xs:choice> <xs:element name= “evtel” type= “xs:string” /> <xs:element name= “istel” type= “xs:string” /> </xs:choice> </xs:complexType> </xs:element>	evtel ve istel isimli elemanlardan sadece biri yer alacaktır. <kişi> <istel>212-3835767</istel > </kişi>
<xs:element name= “kişi”> <xs:complexType> <xs:sequence> <xs:element name= “ad” type= “xs:string” /> <xs:element name= “soyad” type= “xs:string” /> </xs:sequence> </xs:complexType> </xs:element>	ad ve soyad isimli elemanların verilen sırada yer alması önemlidir. <kişi> <ad>Göksel</ad> <soyad>Biricik</soyad> </kişi>

XSD Eleman Tekrarlama Örnekleri

XSD Tanımlamaları	Açıklama / Örnek
<xs:element name= "*kişi*"> <xs:complexType> <xs:sequence> <xs:element name= "*ad*" type= "xs:string" /> <xs:element name= "*cocuk*" type= "xs:string" maxOccurs = "*3*" /> </xs: sequence> </xs:complexType> </xs:element>	<*kişi*> <*ad*>Ahmet</*ad*> <*cocuk*>Ayşe</*soyad*> <*cocuk*>Fatma</*soyad*> </*kişi*>
<xs:element name= "*kişi*"> <xs:complexType> <xs:sequence> <xs:element name= "*ad*" type= "xs:string" /> <xs:element name= "*telefon*" type= "xs:string" minOccurs= "*1*" maxOccurs= "*3*" /> </xs:sequence> </xs:complexType> </xs:element>	<*kişi*> <*ad*>Göksel Biricik>/*ad*> <*telefon*>3835767</*telefon*> <*telefon*>*3835730*</*telefon*> </*kişi*>
<xs:element name= "*kişi*"> <xs:complexType> <xs:sequence> <xs:element name= "*ad*" type= "xs:string" /> <xs:element name= "*ilgi*" type= "xs:string" maxOccurs= "unbounded" /> </xs:sequence> </xs:complexType> </xs:element>	<*kişi*> <*ad*>Göksel Biricik</*ad*> <*ilgi*>Yemek</*ilgi*> <*ilgi*>Tenis</*ilgi*> <*ilgi*>Bisiklet</*ilgi*> ….. </*kişi*>

XSD Eleman Gruplama Örneği

Bazı alt-elemanların belge içinde farklı elemanların altında da aynı özelliklerde kullanıldığı XML belgelerinin geçerliliğini kontrol etmek için XSD'de grup tanımlamalarından yararlanılıp gerektiğinde referans verilir.

- Benzer nitelikte tanım blokları XSD içinde tekrarlanmamış olur
- XSD üzerinde değişiklik yapılması gerekirse tek noktada yapılır.

XSD Tanımlamaları	Açıklama / Örnek
<xs:group name= "bilgiler"> <xs:sequence> <xs:element name= "ad" type= "xs:string" /> <xs:element name= "soyad" type= "xs:string" /> <xs:element name= "tel" type= "xs:decimal" /> </xs:sequence> </xs:group> <xs:element name = "ögretmen" type = "kişibilgileri" /> <xs:complexType name= "kişibilgileri"> <xs:sequence> <xs:group ref = "bilgiler" /> <xs:element name= "email" type= "xs:string" /> </xs:sequence> </xs:complexType>	bilgiler ismiyle tanımlanan grup ad , soyad ve tel isimli elemanlardan oluşmuştur. Tanımlanan grup başka bir noktada group ref="bilgiler" ifadesi ile kullanılabilmektedir. <öğretmen> <ad>Göksel</ad> <soyad>Biricik</soyad> <tel>3835767</tel> <email>gbiricik@yildiz.edu.tr</email> </ öğretmen>

XSD Özellik Gruplama Örneği

Elemanlar gibi özellikler de gruplanabilir.

XSD Tanımlamaları	Açıklama / Örnek
<xs:attributeGroup name= “bilgiler”> <xs:attribute name= “ad” type= “xs:string” /> <xs: attribute name= “soyad” type= “xs:string” /> <xs: attribute name= “tel” type= “xs:decimal” /> </xs:attributeGroup> <xs:element name = “kişi” > <xs:complexType> <xs:attributeGroup ref = “bilgiler” /> </xs:complexType> </xs:element>	<kişi ad=“Göksel soyad= “Biricik” tel= “3835767” />

XSD any ve anyAttribute

Elemanın ve özelliklerin sırası ve tekrarları kontrol edilirken XSD hazırlanırken adları belirlenmemiş/öngörülmemiş eleman/özellikler kullanılabilir. Any ve anyAttribute tanımları burada fayda sağlar.

XSD Tanımlamaları	Açıklama / Örnek
<xs:element name= “kişi”> <xs:complexType> <xs:sequence> <xs:element name= “ad” type= “xs:string” /> <xs:element name= “ilgi” type= “xs:string” /> <xs:any minOccurs= “0” maxOccurs = “3” /> </xs:sequence> </xs:complexType> </xs:element>	Örneğimizde eğitim isimli eleman tanımlanmamış olmasına rağmen xs:any tanımı sayesinde XML belge içinde kullanılabilir. <kişi> <ad>Göksel Biricik</ad> <ilgi>Yemek</ilgi> <eğitim>Doktora</eğitim> </kişi>
<xs:element name= “*kişi*”> <xs:complexType> <xs:sequence> <xs:element name= “*ad*” type= “xs:string” /> <xs:element name= “*ilgi*” type= “xs:string” /> </xs:sequence> <xs:anyAttribute/> </xs:complexType> </xs:element>	*kişi* isimli eleman için herhangi bir öznitelik tanımlanmamış olmasına rağmen xs:anyAttribute tanımı sayesinden *cinsiyet* isimli öznitelik kullanılabilmektedir. *<kişi cinsiyet=“Erkek”>* *<ad>Göksel Biricik</ad>* *<ilgi>Yemek</ilgii>* *</kişi>*

XSD Substitution (Elemanlara yeni isim verme)

Hem önceki isimler, hem de yeni isimler geçerliliğini korur.

XSD Tanımlamaları	Açıklama / Örnek
<xs:element name= “name” type= “xs:string” /> <xs:element name= “ad” substitutionGroup= “name” /> <xs:element name= “instructor” type=“instructorinfo” /> <xs:element name= “öğretmen” substitution=“instructor” /> <xs:complexType name= “instructorinfo”> <xs:sequence> <xs:element ref= “name” /> </xs:sequence> </xs:complexType>	<instructor> <name>Göksel Biricik</name> </instructor> veya <öğretmen> <ad>Göksel Biricik</ad> </öğretmen>

DTD vs XSD

DTD	XSD - Schema
Kendine has yazım kuralları vardır. Bu nedenle geliştirmesi zordur.	XML yazım kurallarına uygun olarak tasarlandığı için geliştirmelere açıktır.
Öğrenilmesi kolay bir standarttır	DTD'ye oranla daha çok şeyin öğrenilmesini gerektirir.
İşlenmesi (herhangi bir nesne modeli kullanarak) olası değildir.	XML üzerine kurulu olması sayesinde DOM (Document Object Model) kullanılarak işlenebilir.
DTD kullanımı için pek çok araç vardır. Ayrıca pek çok uygulamada da kullanılmıştır. Bu nedenle deneyimli kullanıcı sayısı oldukça fazladır.	DTD'den daha sonra standartlaşmış olmak ile birlikte destekleyen pekçok uygulaması vardır ve yaygın olarak kullanılmaktadır.
Özel bir veri tipi tanımı yoktur.	Bilindik veri tiplerinin dışında kullanıcının da amacına uygun veri tipleri tanımlamasına imkan verir.
Genel olarak yapının (structure) kontrolü için yeterlidir. Ancak içeriğin kontrolü üzerinde hiçbir etkisi yoktur.	Yapı üzerinde yer alan elemanların gerek diziliş, gerek tekrar, gerek ise içerik ve kendi aralarındaki ilişkilere dayalı pek çok kontrolün yapılmasına imkan verir.
İsim uzayı (namespace) desteği yoktur.	İsim uzayı (namespace) desteği vardır.

Gelecek Ders

XSL, XSLT, XPath