XML, XSD, Namespace, URI ve URL Kavramları

Extensible Markup Language (XML)

- XML, **verinin anlamı**nın/yapısının verinin kendisi ile birlikte taşınmasını sağlar
- Veri **transferi ve saklama** da kullanılır
- XML sadece **veri hakkında bir bilgi**yi ifade eder
- Bunun dışında o veri ile ilgili **başka herhangi bir şey** yapmaz
- Veriye özel XML **elemanları** ve **attribute**'lar kullanılarak XML mesajları oluşturulur

XML

- Verinin oluşturulması, sistemler arası transferi veya veritabanına kaydedilmesi, verinin gösterimi **farklı farklı sistemler** tarafından ele alınan farklı farklı işlemlerdir
- Verinin XML formatında olması bunlarla ilgili **herhangi bir kabiliyet**in olması/sergilenmesi anlamına gelmez
- XML sadece **verinin ne olduğu**na odaklanır
- **HTML** ise **verinin gösterimi**ne odaklanır

XML Örneği

```
<vets>

   <vet id="101">
      <firstName>Ali</firstName>
      <lastName>Zor</lastName>
      <graduationYear>2011</graduationYear>
   </vet>

   <vet id="102">
      <firstName>Veli</firstName>
      <lastName>Uysal</lastName>
      <graduationYear>2005</graduationYear>
   </vet>

</vets>
```

XML Namespace Nedir?

```
<table>
  <tr>
    <td>Elma</td>
    <td>Armut</td>
  </tr>
</table>
```

HTML sayfa içerisindeki table elemanıdır

```
<table>
   <name>Yemek Masasi</name>
   <width>80</width>
   <length>120</length>
</table>
```

Bir masa bilgisini ifade eden table elemanıdır

```
<data>
    <table>
        <tr>
            <td>Elma</td>
            <td>Armut</td>
        </tr>
    </table>
</data>
```

```
<data>
    <table>
        <name>Yemek Masasi</name>
        <width>80</width>
        <length>120</length>
    </table>
</data>
```

Problem:
İki farklı table yapısının aynı mesajda ele alınması karışıklıklara yol açacaktır
Hangisi HTML table, hangisi masayı ifade eden table net değildir

Çözüm:
XML namespace tanımlarını kullanmaktır

XML Namespace Nedir?

- XML mesajındaki eleman ve attribute'ları **benzersiz biçimde ayrıştıran** bir URI'dır
- Genellikle kök elemanda tanımlanır

```
<data
xmlns:html="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com">
    <html:table>
        <html:tr>
            <html:td>Elma</html:td>
            <html:td>Armut</html:td>
        </html:tr>
    </html:table>
    <furniture:table>
        <furniture:name>Yemek Masasi</furniture:name>
        <furniture:width>80</furniture:width>
        <furniture:length>120</furniture:length>
    </furniture:table>
</data>
```

xmlns:prefix="URI" ile tanımlanır. Prefix sayesinde namespace'i uzun biçimde elemanların başına yazmaya gerek kalmaz

Default XML Namespace

- XML içindeki namespace'lerden bir tanesi **default** olarak tanımlanabilir

Default bir namespace tanımı yapılabilir
default namespace'in elemanları
prefix olmadan da kullanılabilir

```
<data
xmlns="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com">
    <table>
        <tr>
            <td>Elma</td>
            <td>Armut</td>
        </tr>
    </table>
    <furniture:table>
        <furniture:name>Yemek Masasi</furniture:name>
        <furniture:width>80</furniture:width>
        <furniture:length>120</furniture:length>
    </furniture:table>
</data>
```

XML Şema (XSD) Nedir?

- XML Şema bir **XML dokümanın/mesajın yapısı**nı tanımlar
- **XSD** olarak da bilinir
- Doküman içerisinde yer alacak **eleman** ve **attribute**'ların neler olabileceğini, bunlar arasındaki yapısal ilişkileri belirtir
- Eleman ve attribute'ların **veri tipleri**ni, hangi sabit değerleri alabileceklerini veya default değerlerin neler olabileceğini belirtir

XML Şema (XSD) Nedir?

- Bir elemanın **çocuk elemanlar**ının neler olabileceğini belirtir
- Çocuk elemanların **sayısı, sırası** tanımlanabilir
- Bir elemanın **içeriğinin olup olamayacağı**nı, text değer içerip içeremeyeceğini belirtir

XML Şema Örneği

```
<schema ...>

    <complexType name="vet">
        <sequence>
            <element name="firstName" minOccurs="1" maxOccurs="1"
type="string"/>
            <element name="lastName" minOccurs="1" maxOccurs="1"
type="string"/>
            <element name="graduationYear" minOccurs="1" maxOccurs="1"
type="integer"/>
            <element name="specialty" minOccurs="0"
maxOccurs="unbounded" type="ws:specialty"/>
        </sequence>
    </complexType>

    <complexType name="specialty">
        <sequence>
            <element name="name" minOccurs="1" maxOccurs="1"
type="string"/>
        </sequence>
    </complexType>

</schema>
```

XSD Tanımının Yapısı

Bir XSD tanımının kök elemanı her zaman için <schema>'dır

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.java-egitimleri.com/vets"
    xmlns="http://www.java-egitimleri.com/vets"
    elementFormDefault="qualified">
...
</schema>
```

Şema içerisinde kullanılacak eleman ve veri tiplerinin XMLSchema namespace ve ön eki tanımlanır.

Bu şema tarafından tanımlanacak XML elemanlarının ait olacakları namespace'i tanımlar

Target namespace ve ön eki tanımlanır

Burada tanımlanmış elemanların herhangi bir XML dokümanı içerisinde kullanılması durumunda namespace qualifikasyonuna sahip olmaları gerektiğini anlatır

XML Dokümanlarının Yapısı

- Bir XML dokümanını oluşturan **temel yapı taşları:**
  - Element
  - Attribute
  - PCDATA
  - CDATA
  - Entity

XML Dokümanlarının Yapısı

- **Element**, XML dokümanların ana yapı taşıdır
- İçerisinde **text değer** yer alabilir
- **Çocuk elemanlar** yer alabilir veya herhangi bir şey de olmayabilir
- **Attribute** ise element'ler hakkında **ilave bilgi** taşımayı sağlar
- Bir attribute her zaman için bir XML elemanının **başlangıç tag**'i içerisinde yer alır

Simple Element

- **Simple Element**, sadece text veri içerebilecek eleman demektir
- İçerisinde **çocuk eleman** veya **attribute olamaz**
- Text **verinin tipi** tanımlanabilir, veri üzerinde **kısıtlama** yapılabilir veya verinin belirli bir **pattern**'a uyması istenebilir
- Yaygın kullanılan **veri tipleri**: string, decimal, integer, boolean, date

Simple Element Örnekleri

```
<element name="firstName" type="string"/>

<element name="lastName" type="string"/>

<element name="graduationYear" type="integer"/>

<element name="color" type="string" default="red"/>

<element name="color" type="string" fixed="red"/>
```

Fixed veya default değerler de tanımlanabilir

Complex Element

- İçerisinde **çocuk eleman** veya **attribute** barındırabilen eleman'lardır
- İçerisinde herhangi bir şey de içermeyebilir, buna **complex empty element** denir
- **Sadece attribute** içerebilir
- Genellikle complex element ya çocuk eleman, ya da text içerik barındırır
- Hem çocuk eleman, hem de text içeriği aynı anda barındıracak ise buna **mixed content** adı verilir

Complex Element Örneği

```
<element name="vet">

    <complexType>

        <sequence>
            <element name="firstName" type="string" />
            <element name="lastName" type="string" />
            <element name="graduationYear" type="integer" />
        </sequence>
    </complexType>
</element>
```

<complexType> elemanı ile tanımlanır

Çocuk elemanların belirtilen sırada olması gerektiğini tanımlar

Çocuk elemanlar

Sequence dışında başka XSD indicator'leri de vardır:
all: çocuk elemanlar herhangi bir sırada olabilir, fakat Her biri sadece bir kere yer alabilir
choice: Çocuk elemanlardan sadece bir tanesi olabilir

Complex Element Örneği

Complex element tanımlarken diğer bir yöntemde type tanımını elemanın Dışında bir yerde yapmaktır. Buna global type adı verilir.

```
<element name="vet" type="tns:vetType"/>

<complexType name="vetType">
    <sequence>
        <element name="firstName" type="string" />
        <element name="lastName" type="string" />
        <element name="graduationYear" type="integer" />
        <element name="specialty"   type="tns:specialtyType"
                                    minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="specialtyType">

    <simpleContent>

        <extension base="string"/>
    </simpleContent>
</complexType>
```

Elemanın kaç defa tekrarlayabileceğini belirtir

Complex element'in sadece text içerik barındırmasını sağlar

Attribute ve Mixed Content

```
<complexType name="vetType">

    <attribute name="id" type="string"/>

    <sequence>
        <element name="firstName" type="string" />
        <element name="lastName" type="string" />
        <element name="graduationYear" type="integer" />
    </sequence>
</complexType>
```

<attribute> elemanı ile attribute tanımlanır. İsmi, tipi, default değeri vs belirtilir

```
<xs:element name="letter">
    <xs:complexType mixed="true">
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="orderid" type="xs:positiveInteger" />
            <xs:element name="shipdate" type="xs:date" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element içerisinde hem çocuk elemanların Hem de text içeriğin birlikte olabileceğini belirtir

Ne Zaman Element, Ne Zaman Attribute Kullanmalı ?

- Attribute'lar **çoklu değer** içeremezler
- XSD ve DTD tarafından **denetlenmeleri** de zor olabilir
- Eğer attribute'da tutulacak değer **veri ile doğrudan ilgili** ise bir element ile ifade etmek daha doğrudur
- Aksi takdirde attribute kullanılabilir
- **ID** attribute değeri bu kurala bir istisna olabilir

XML Dokümanlarının Yapısı

- **PCDATA**, XML parser tarafından parse edilen **text içerik**tir
- Text içerik içerisinde yer alan tag'ler veya entity karakterler **parser tarafından işleme tabi tutulur**
- **CDATA** ise XML parser tarafından herhangi bir işleme tabi tutulmayan text içeriktir

XML Dokümanlarının Yapısı

- **Entity** ise XML dokümanı içerisinde yer alan ve XML için **özel anlam içeren karakter**lerin ifade edilmesini sağlar
- Örneğin “<” işareti bir XML elemanın tag'nin ifade edilmesinde kullanılır
- Entity gösterimi ile özel karakter **XML dokümanı içerisinde** herhangi bir problem çıkarmadan kullanılabilir

Ön Tanımlı Entity'ler

Entity	Karakter
&lt;	<
&gt;	>
&amp;	&
&quot;	“
&apos;	'

XML Dosyadan XSD'ye Referans

- XML içerisinde yapısını tanımlayan XSD'nin lokasyon bilgisi de belirtilebilir

xmlns:xsi genellikle XML ve XSD dosyalarının içerisinde kullanılan built-in attribute'lara ait namespace tanımıdır

```
<data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:html="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com"

xsi:schemaLocation="
http://www.w3.org/1999/xhtml http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd
http://www.furnitureworld.com furnitures.xsd">

...

</data>
```

XML dokümanının yapısını tanımlayan şema dosyalarının lokasyonu da built-in **schemaLocation** ile belirtilir. Böylece XML **parser** veya **editor** XML dokümanını XSD'ye göre **validate** etme imkanına kavuşur

XSD mi? DTD mi?

- **XSD öncesi**nde XML dokümanların yapısını tanımlamak için **DTD** kullanılırdı
- Yeni nesil XML dokümanlarında artık **XSD tercih edilmekte**dir
- DTD bir XML dokümanının içerisinde veya harici başka bir dosyada tanımlanabilir
- XSD ise **tamamen ayrı bir dosya** da tanımlanmalıdır

XSD mi? DTD mi?

- XML dokümanının yapısını tanımlayan XSD ve DTD'ler **XML doküman içerisinden refer** edilirler
- XSD veya DTD sayesinde XML dokümanların içeriklerinin **uygun yapıda** olup olmadığı, **geçerli** ve olması gereken veriyi içerip içermediği kontrol edilebilir
- Farklı gruplar veya sistemler birbirleri ile XML dokümanlarını **standart biçimde paylaşma** imkanı bulurlar

DTD Örneği

```
<!ELEMENT vets(vet*)>
<!ELEMENT vet(firstName,lastName,graduationYear)>
<!ELEMENT firstName(#PCDATA)>
<!ELEMENT lastName(#PCDATA)>
<!ELEMENT graduationYear(#CDATA)>
<!ATTLIST vet id ID "0">
```

*: sıfır veya daha fazla, +: bir veya dafa fazla anlamındadır

Default değerdir. #REQUIRED, #IMPLIED, #FIXED *value* gibi değerler de olabilir

Attribute ismidir

Attribute'un tipini belirtir CDATA,ID,IDREF,ENTITY,NMTOKEN, IDREFS,ENTITIES,NMTOKENS, NOTATION, xml:*value* gibi değerler de olabilir

XML Dokümanından DTD'ye Referans

JAVA Eğitimleri

```
<?xml version="1.0" encoding="UTF-8"?>
```

Kök eleman

DTD'yi bir uygulamaya özel benzersiz biçimde tanıtır
PUBLIC ise birden fazla uygulama için bunu yapar

```
<!DOCTYPE vets SYSTEM "vets.dtd">
```

DTD'nin URI referansıdır, DTD'nin lokasyonunu belirtir

```
<vets>
    <vet id="101">
        <firstName>Ali</firstName>
        <lastName>Zor</lastName>
        <graduationYear>2011</graduationYear>
    </vet>

    <vet id="102">
        <firstName>Veli</firstName>
        <lastName>Uysal</lastName>
        <graduationYear>2005</graduationYear>
    </vet>
</vets>
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com