

XML

Extensible Markup Language

Yrd. Doç. Dr. Özgü Can

XML

- eXtensible Markup Language
- Microsoft tarafından .NET ortamında kullanılmak için geliştirilmemiş olmasına rağmen, .NET'deki bir çok uygulamada kullanılmaktadır.
 - .config dosyaları
 - Web servisleri arasında bilginin transfer edilmesi

XML

- W3C (World Wide Web Consortium, <http://www.w3.org>) organizasyonu tarafından tanımlanmış bir standarttır.
- XML'de, veri basit bir metin (text) biçiminde saklanmaktadır.
 - Böylelikle, veri herhangi bir bilgisayar tarafından okunabilir.

XML

- XML, .NET ortamında veri iletiminde varsayılan biçim olduğundan önemlidir.
- Visual Studio, XML ile ilgili bir çok işlem ile ilgilenmektedir.

XML Belgesi

- XML'de yer alan bütün veri kümesi **XML belgesi (XML document)** olarak adlandırılmaktadır.
- XML belgesi;
 - Belirli kurallara uymalıdır.
 - Farklı kısımlardan oluşur.
 - XML ögesi
 - Belgenin mevcut verisinden oluşur.

XML Öğesi

- XML öğesi (XML element), **açılmış** ve **kapanmış etiketlerden (tag)** oluşur.

ÖR:

<book> **Açma etiketi**

</book> **Kapama etiketi**

- Bu etiketlerin arasında **veri** yer alır.

ÖR:

<book>1984</book>

XML Öğesi

- XML öğesinin **büyük-küçük harf duyarlılığı** (case-sensitive) vardır.

ÖR:

Aynı değildir!

<book> ↔ <BOOK>

<book>

1984

</BOOK>

Geçersiz XML

XML Ayırıştırıcı (XML Parser)

- XML öğelerini **inceleyerek**, XML belgelerini **okuyan** ve bu belgeleri **analiz** eden programlardır.
- Geçersiz XML içeren belgeleri reddeder.

XML Öğesi

- XML öğeleri, diğer öğeleri içerebilir.

ÖR:

```
<book>
```

```
  <title> 1984 </title>
```

```
  <author> George Orwell </author>
```

```
</book>
```

XML Öğesi

- Örtüşen (overlapping) öğelere izin verilmemektedir.
- Bu nedenle, üst (parent) öğenin kapatılmasından önce bütün alt-öğeler kapatılmalıdır.

ÖR:

```
<book>
```

```
  <title> 1984
```

```
  <author> George Orwell
```

```
  </title> </author>
```

```
</book>
```

Geçersiz XML

XML Öğesi

- İçerisinde veri olmayan **boş öğeler** olabilir.

ÖR:

```
<book> </book>
```

- Boş öğeler için **kısaltılmış söz dizimi** kullanılabilir.

ÖR:

```
<book />
```

XML Öznitelikleri (XML Attributes)

- Veri, **özniteliklerin** içerisinde de saklanabilir.

- Öznitelikler:

ad = "değer"

biçimindedir.

XML Öznitelikleri (XML Attributes)

ÖR:

`<book title="1984"> </book>`

ya da

`<book title='1984'> </book>`

`<book title= 1984> </book>`



Geçersiz XML

XML Öğeleri vs. XML Öznitelikleri

```
<book title="1984"> </book>
```

ve

```
<book>
```


```
  <title> 1984 </title>
```

```
</book>
```

arasında fark var mıdır?

Bir biçimin diğer biçime göre herhangi bir avantajı yoktur.

XML Öğeleri vs. XML Öznitelikleri

- **Öğeler ile ilgili alt bilgi eklenecekse** öğelerin kullanımı daha iyi bir seçimdir.
 - Bir öğeye **alt-öğeler** ve **öznitelikler** eklenebilir.
 Öznitelikler için bu gerçekleştirilemez.
- Öğeler daha **okunaklı** ve **düzenlidir**.

XML Öğeleri vs. XML Öznitelikleri

- Veri, ağ üzerinden **sıkıştırılmamış** olarak gönderilecekse, **öznitelikler için** daha az bant genişliği harcanmaktadır.
- Öznitelikler, belge ile ilgili **her kullanıcıyı ilgilendirmeyen** **bilginin tutulması** için uygundur.

XML Öğeleri vs. XML Öznitelikleri

- Öğelerin ve özniteliklerin her ikisi de kullanılabilir.
- Hangisinin kullanılacağı **kişinin tercihin**e bağlıdır.

XML Bildirimi (XML Declaration)

- XML belgeleri çeşitli bölümleri içerebilir.
- Birbirinden bağımsız bu bölümler **düğüm (node)** olarak adlandırılmaktadır.
- Öğeler, öğelerin içerisindeki metin ve öznitelikler XML belgesinin bütün düğümleridir.

XML Bildirimi

- Bütün XML belgelerinde yer alan düğüm XML bildirimleridir.
- XML bildirimi, belgenin ilk düğümünde yer almalıdır.

XML Bildirimi

ÖR: XML bildirimi:

```
<?xml version="1.0"?>
```

- XML bildiriminde:
 - ?, **xml** adı, **version** özniteliği mutlaka yer alır.
 - Versiyon **1.0** ya da **1.1** olabilir.
 - Visual Studio, **1.1** versiyonunu desteklemektedir.
 - W3C, mümkün olduğunca **1.1** versiyonunun kullanılmasını teşvik etmektedir.

XML Bildirimi

- Seçimli olarak;

- **encoding**

- Belgeyi okumak için kullanılacak karakter setini belirtir.
 - ÖR: **UTF-16** → **16 bit unicode karakter seti**

- **standalone**

- XML belgesinin herhangi başka dosyalara bağlı olup olmadığını belirtir.
 - **yes** ya da **no** değerlerinden birini alır.

kullanılabilir.

- Çoğunlukla, sadece **version** özniteliği kullanılır.

XML Belgesinin Yapısı

- Geleneksel veritabanında veriler tablolarda tutulmaktadır.
- Tablolarda veriler, satırlar ve sütunlarda saklanmaktadır.
- XML'de veri **hiyerarşik** bir yapıdadır.
 - ÖR: Windows Explorer

XML Belgesinin Yapısı

- Her bir XML belgesinin tek bir **kök ögesi (root element)** vardır.
 - Bütün öğeler ve metin verisi kök öğenin içerisinde yer almaktadır.

XML Belgesinin Yapısı

- Belgenin üst seviyesinde birden fazla öge olursa → **Belge geçerli bir XML değildir.**
- Diğer XML düğümleri üst seviyede yer alabilir.

XML Belgesinin Yapısı

Geçerli XML

```
<?xml version="1.0"?>
<books>
  <book>1984</book>
  <book>Animal Farm</book>
  <book>Burmese Days</book>
</books>
```

Geçersiz XML

```
<?xml version="1.0"?>
<book>1984</book>
<book>Animal Farm</book>
<book>Burmese Days</book>
```

XML Ad Uzayı (XML Namespace)

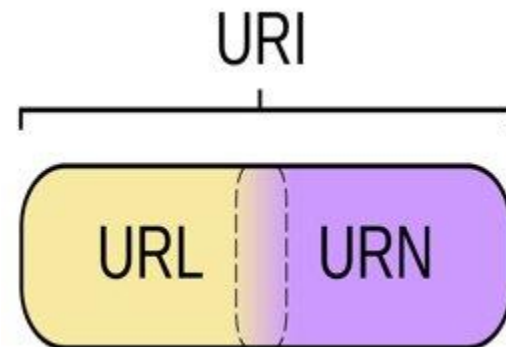
- **XML ad uzayı**, XML söz varlıklarını (vocabulary) tanımlamak için kullanılır.
- XML kaynakları **aynı isimde etiketlere** sahip olabilir.
- Farklı söz varlıklarında yer alan öğelerin, tek bir XML belgesinde kullanımı sağlanır.
 - Öğelerin **yanlış yorumlanması (misinterpreting)** riski önlenmiş olur.

XML Ad Uzayı (XML Namespace)

- Belirli öğeler ya da öznitelikler, belirli bir ad uzayı ile **ön ek (prefix)** kullanılarak ilişkilendirilmektedir.
- Her ad uzayının **tek/benzersiz (unique)** olduğu garantilenmelidir.
 - Bunu sağlamak için ön ekler, **tek/benzersiz olan URI**'ler ile ilişkilendirilmektedir.

XML Ad Uzayı (XML Namespace)

- **U**niform **R**esource **I**dentifier – URI
 - İnternetteki bir kaynağı ya da adı tanımlamak için kullanılmaktadır.
- İki şekilde olabilir:
 - **U**niform **R**esource **L**ocator – URL
 - **U**niform **R**esource **N**ame – URN



XML Ad Uzayı (XML Namespace)

- **U**niform **R**esource **L**ocator – URL
 - Kaynağın **konumunu** belirtir.
 - URL, **protokol adını** içermektedir.
 - Her URL bir URI'dir, ancak tersi geçerli değildir.

ÖR:

http://egeweb2.ege.edu.tr/bilmuhdp/course_plan.php

ftp://ftp.bilmuh.ege.edu.tr/bilmuh/course_plan.doc

XML Ad Uzayı (XML Namespace)

- Uniform Resource Name – URN
 - Kaynağın kesin olarak belirtilmesi şeklidir.
 - Konumdan bağımsızdır.
 - Protokol içermez.
 - Her URN bir URI'dir, ancak tersi geçerli değildir.

ÖR:

urn:isbn:0451450236

XML Ad Uzayı (XML Namespace)

- Ad uzayı ön eki, bir URI ile ilişkilendirilir.
- Ön eki, belirli bir ad uzayı ile ilişkilendirmek için;
 - Öğenin içerisinde **xmlns:prefix** özniteliği kullanılmaktadır.
 - Bu özniteliğin değeri, ad uzayını belirten **tek bir URI** olarak belirtilmektedir.
 - Ön ek, öge içerisinde herhangi bir yerde kullanılabilir.

XML Ad Uzayı (XML Namespace)

- Ad uzayı, web üzerinde bir dokümanı ya da adresi göstermez.
- Ön ek, etiketlerin başında kullanılarak **aynı isimdeki etiketlerin hangi ad uzayına ait olduğunu** gösterir ve **çakışmasını engeller**.

XML Ad Uzayı (XML Namespace)

ÖR:

```
<?xml version="1.0"?>
```

```
<books>
```

```
  <book xmlns:can="http://www.canyayinlari.com">
```

```
    <can:title>Animal Farm</can:title>
```

```
    <can:author>George Orwell</can:author>
```

```
  </book>
```

```
</books>
```

✓ **can:** ön eki, **<title>** ve **<author>** öğeleri
<book> öğesi içerisinde yer aldıklarından
bu öğeler ile birlikte kullanılabilir.



can: ön eki, **<books>** öğesi için
tanımlanmadığından **<books>** öğesine
eklenemez.

XML Ad Uzayı (XML Namespace)

- Öğesi için varsayılan ad uzayı tanımlanabilir.

`<book>` öğesi için `http://www.canyayinlari.com`

varsayılan ad uzayıdır.

```
<?xml version="1.0"?>
```

Bütün öğeler bu ad uzayına aittir.

```
<books>
```

```
  <book xmlns="http://www.canyayinlari.com">
```

```
    <title>Animal Farm</title>
```

```
    <author>George Orwell</author>
```

```
    <html:img alt="Cover Image" src="animalfarm.gif"
```

```
    xmlns:html="http://www.w3.org/1999/xhtml" />
```

```
  </book>
```

```
</books>
```

Öğesi için farklı bir ad uzayı tanımlanmak istenirse

İyi Biçimlendirilmiş ve Geçerli XML

- Legal XML iki biçimde olabilir:
 - İyi-biçimlendirilmiş (well-formed)
 - Geçerli (valid)

İyi Biçimlendirilmiş ve Geçerli XML

- İyi-biçimlendirilmiş
 - XML standardının **bütün kurallarına** uyan belgelerdir.
- Eğer, belge iyi-biçimlendirilmemiş ise XML ayrıştırıcılar (parser) belgeyi **yorumlayamaz** ve belgeyi **reddeder**.

İyi Biçimlendirilmiş ve Geçerli XML

- İyi-biçimlendirilmiş bir belgede:
 - Sadece **tek bir kök ögesi** (root element) bulunur.
 - Her bir öge için **kapama etiket**lerine sahiptir.
 - **Örtüşen öğeler** yoktur.
 - Bütün alt (child) öğeler üst (parent) öğenin içerisinde yer alır.
 - **Bütün öznitelikler** tırnak işareti ile çevrelenmiştir.

İyi Biçimlendirilmiş ve Geçerli XML

- XML belgesi bütün kurallara uyduğu halde, geçerli olmayabilir.
- XML, bir dil değil, **XML uygulamalarının tanımlanması** için bir standarttır.

İyi Biçimlendirilmiş ve Geçerli XML

- İyi-biçimlendirilmiş XML, XML standardına uyar.
- Geçerli XML, aynı zamanda, XML uygulamaları için tanımlanmış kurallara da uymalıdır.
 - Bütün ayrıştırıcılar belgelerin geçerliliğini kontrol etmez.
 - Geçerliliği kontrol eden ayrıştırıcılara **geçerliliği denetleyen ayrıştırıcılar (validating parsers)** denmektedir.

XML Belgelerinin Geçerliliğinin Denetlenmesi

XML, belge içerisinde;

- Hangi öğelerin ve özniteliklerin

ve

- Hangi sırada

yer alabileceğini tanımlamayı iki şekilde desteklemektedir:

- Belge Türü Tanımları (Document Type Definitions, DTDs)

ve

- Şemalar (Schemas)

Belge Türü Tanımı (DTD)

- DTD, öğelerin ve özniteliklerin veri türlerini tanımlamanıza izin vermediği için kullanımı esnek değildir.
- .NET Framework bağlamı içerisinde;
 - DTD **çok sık** kullanılmamaktadır.
 - XML uyumlu söz dizimi kullanmaz.
 - Şemalar **daha sık** kullanılmaktadır.
 - XML uyumlu söz dizimi ile yazılırlar.
 - Veri türleri belirtilmesine izin verirler.
 - Karmaşıktır.
 - Tanımlanmasında çok çeşitli biçimler vardır.

Şemalar (Schemas)

- Şemalar için .NET'de iki farklı biçim desteklenmektedir:
 1. XML Schema Definition Language (XSD)
 2. XML Data Reduced (XDR)

XDR Şemaları

- Eski bir Microsoft standardıdır.
- Microsoft-dışı ayrıştırıcılar tarafından kullanılmamaktadır.

XSD

- W3C tarafından önerilen açık bir standarttır.
 - <http://www.w3schools.com/schema/>
- Şemalar, XML belgesinin içerisinde yer alabileceği gibi ayrı bir dosyada da tutulabilir.

Book.xml

```
<?xml version="1.0"?>
<books>
  <book>
    <title>Nineteen Eighty-Four</title>
    <author>George Orwell</author>
    <code>7582</code>
  </book>
  <book>
    <title>In Cold Blood</title>
    <author>Truman Capote</author>
    <code>7043</code>
  </book>
</books>
```

XSD Şeması

- XSD şeması içerisindeki öğeler <http://www.w3.org/2001/XMLSchema> ad uzayına ait olmalıdır.
- Bu ad uzayı eklenmezse, şema öğeleri tanınmayacaktır.

XSD Şeması

- XML belgesini başka bir dosyadaki XSD ile ilişkilendirmek için:
 - **schemalocation** ögesi kök ögesine eklenmektedir.

```
<?xml version="1.0"?>  
<books schemalocation="file://C:\WindowsProgramming\XML\books.xsd">  
.....  
</books>
```

XSD Şeması - Örnek

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="books">
    <complexType>
      <choice maxOccurs="unbounded">
        <element name="book">
          <complexType>
            <sequence>
              <element name="title" />
              <element name="author" />
              <element name="code" />
            </sequence>
          </complexType>
        </element>
      </choice>
      <attribute name="schemaLocation" />
    </complexType>
  </element>
</schema>
```


XSD Şeması

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
```

- Varsayılan ad uzayı, XSD ad uzayı olarak belirtilmiştir.
- Böylelikle, ayrıştırıcıya (parser), belgedeki bütün öğelerin şemaya ait olduğu belirtilmiş olur.
- Ad uzayı belirtilmez ise ayrıştırıcı;
 - Öğelerin normal XML öğeleri olduğunu varsayacaktır.
 - Geçerliliğini onaylamak için bu öğelere ihtiyacı olduğunu fark etmeyecektir.

XSD Şeması

- Bütün şema **<schema>** ögesi içerisinde.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="books">
    <complexType>
      <choice maxOccurs="unbounded">
        <element name="book">
          <complexType>
            <sequence>
              <element name="title" />
              <element name="author" />
              <element name="code" />
            </sequence>
          </complexType>
        </element>
      </choice>
      <attribute name="schemaLocation" />
    </complexType>
  </element>
</schema>
```

XSD Şeması

- Belge içerisinde yer alacak olan her bir öge **<element>** etiketi ile belirtilmelidir.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="books">
    <complexType>
      <choice maxOccurs="unbounded">
        <element name="book">
          <complexType>
            <sequence>
              <element name="title" />
              <element name="author" />
              <element name="code" />
            </sequence>
          </complexType>
        </element>
      </choice>
      <attribute name="schemaLocation" />
    </complexType>
  </element>
</schema>
```

XSD Şeması

- **<element>** etiketi, öğenin adını belirten **name** özneliğine sahiptir.

```
<element name="book">
```

```
<element name="title" />
```

```
<element name="author" />
```

```
<element name="code" />
```

XSD Şeması

- Öğenin, iç içe alt öğeleri varsa;
 - **<element>** etiketi **<complexType>** öğesinin içerisinde yer alır.

```
<element name="book">
  <complexType>
    <sequence>
      <element name="title" />
      <element name="author" />
      <element name="code" />
    </sequence>
  </complexType>
</element>
```

<choice>* öğesi, alt öğelerden sadece birinin kapsayan öğe içerisinde gösterilebileceğini belirtir.

```
<choice maxOccurs="unbounded">
  <element name="book">
    <complexType>
      <sequence>
        <element name="title" />
        <element name="author" />
        <element name="code" />
      </sequence>
    </complexType>
  </element>
</choice>
```

Attribute	Description
id	Optional. Specifies a unique ID for the element
maxOccurs	Optional. Specifies the maximum number of times the choice element can occur in the parent element. The value can be any number ≥ 0 , or if you want to set no limit on the maximum number, use the value "unbounded". Default value is 1
minOccurs	Optional. Specifies the minimum number of times the choice element can occur in the parent the element. The value can be any number ≥ 0 . Default value is 1
any attributes	Optional. Specifies any other attributes with non-schema namespace

* http://www.w3schools.com/schema/el_choice.asp

<sequence>* öğesi, alt öğelerin sıra ile belirtilmesi gerektiğini belirtir.

Öğesi birden fazla görünecekse

```
<choice maxOccurs="unbounded">
  <element name="book">
    <complexType>
      <sequence>
        <element name="title" />
        <element name="author" />
        <element name="code" />
      </sequence>
    </complexType>
  </element>
</choice>
```

Öğesi limitsiz olarak bulunabilir.

Attribute	Description
id	Optional. Specifies a unique ID for the element
maxOccurs	Optional. Specifies the maximum number of times the sequence element can occur in the parent element. The value can be any number ≥ 0 , or if you want to set no limit on the maximum number, use the value "unbounded". Default value is 1
minOccurs	Optional. Specifies the minimum number of times the sequence element can occur in the parent element. The value can be any number ≥ 0 . Default value is 1
any attributes	Optional. Specifies any other attributes with non-schema namespace

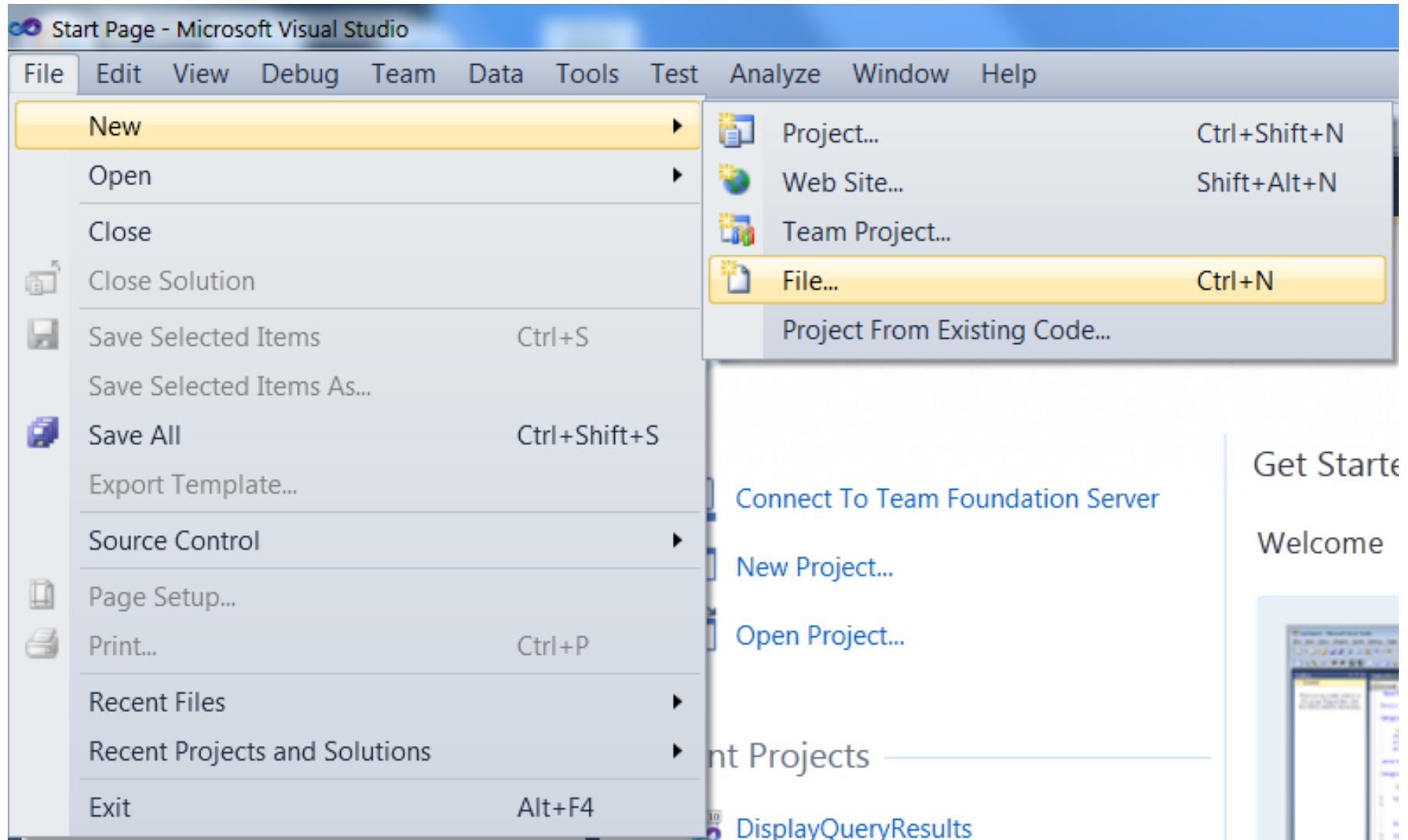
* http://www.w3schools.com/schema/el_sequence.asp

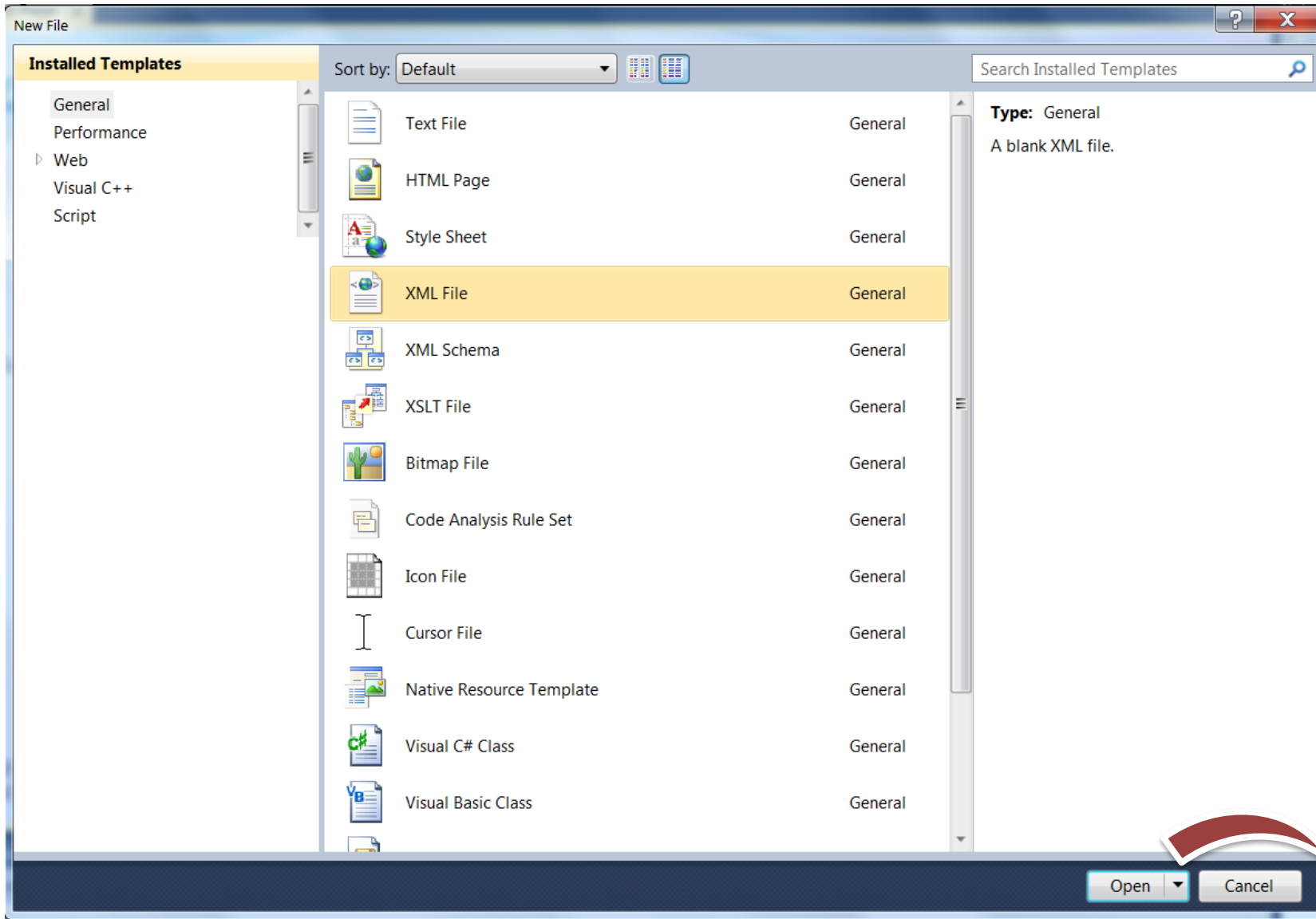
XSD Şeması

- Öznitelikler, **<attribute>** öğeleri ile temsil edilmektedir.
- **schemalocation** özneliği ayrıştırıcıya, şemayı nerede bulacağını belirtmektedir.

```
<attribute name="schemalocation" />
```

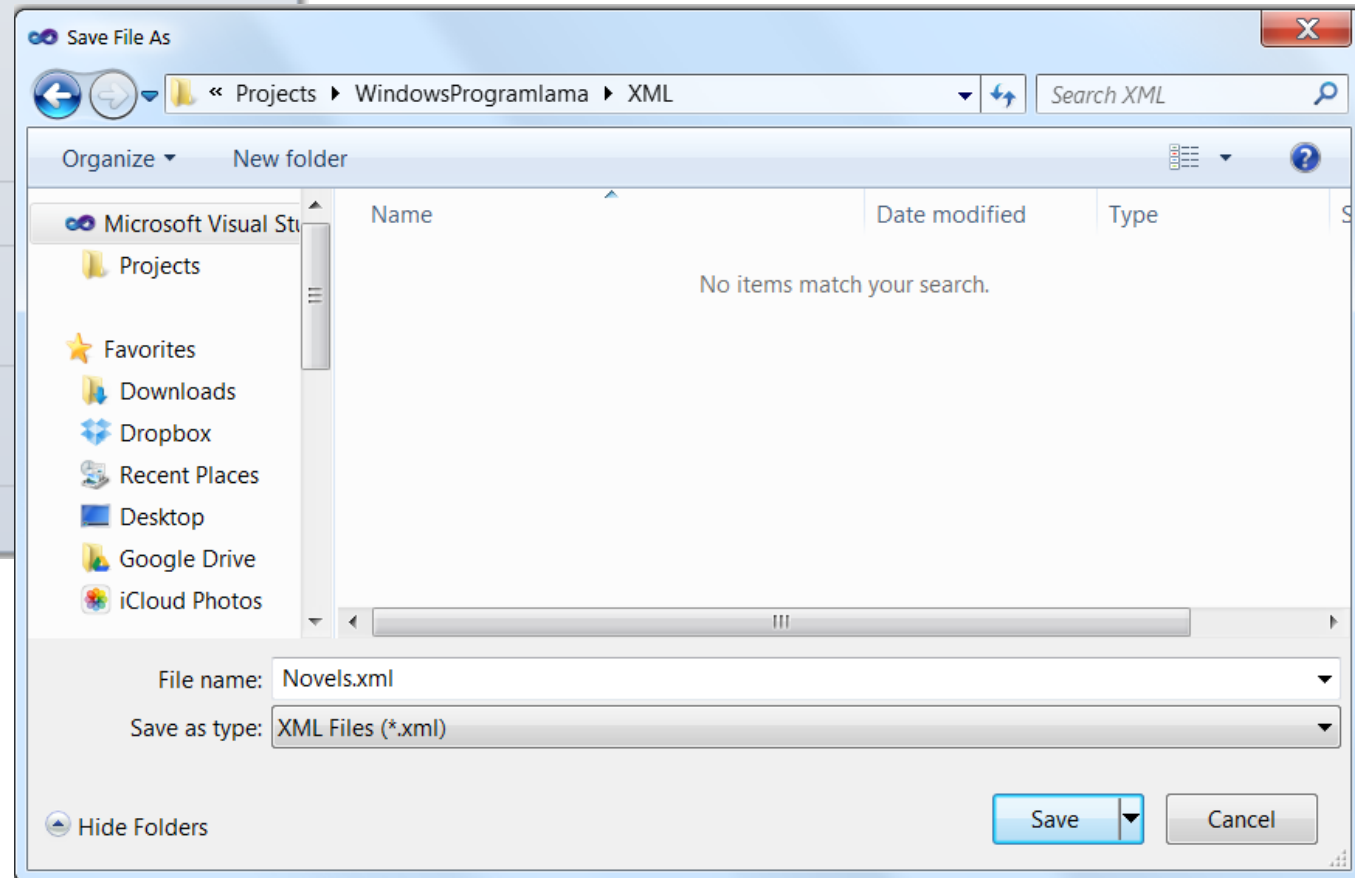
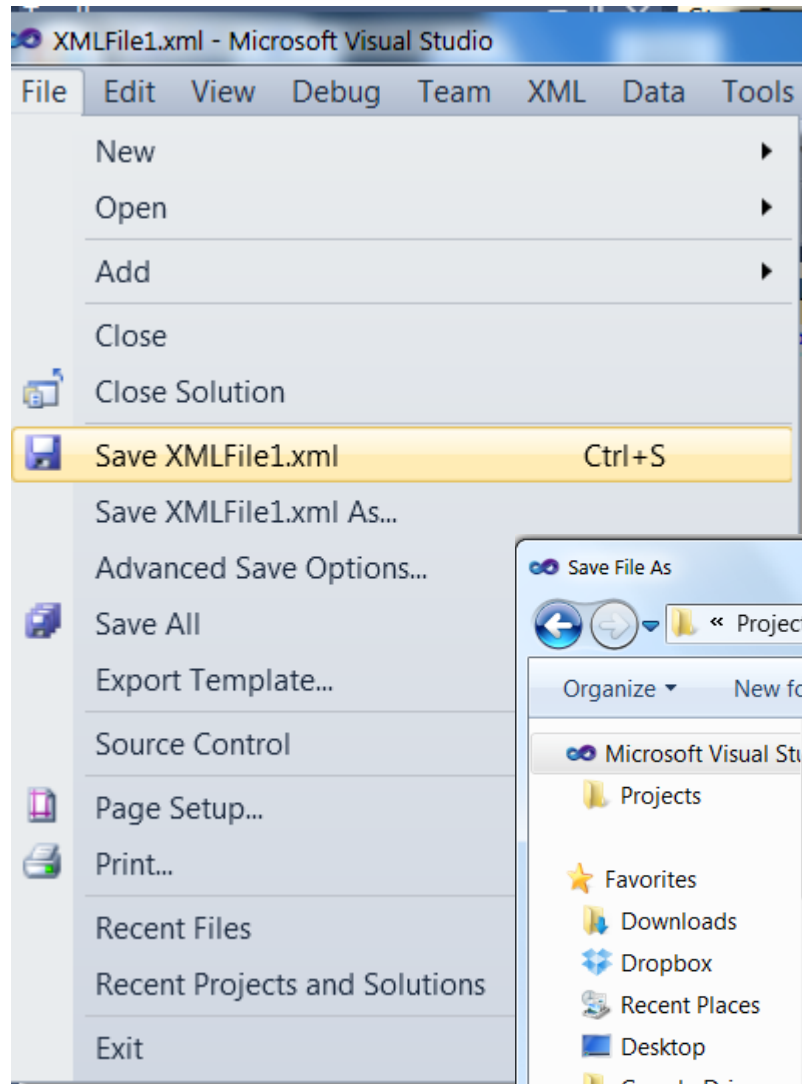

Uygulama - XML





The screenshot shows the XML code in the editor. The code is: `<?xml version="1.0" encoding="utf-8"?>`. The code is color-coded: `<` is blue, `?` is red, `xml` is blue, `version="1.0"` is blue, `encoding="utf-8"?>` is red. A red arrow points from the 'Open' button in the dialog box above to the code.

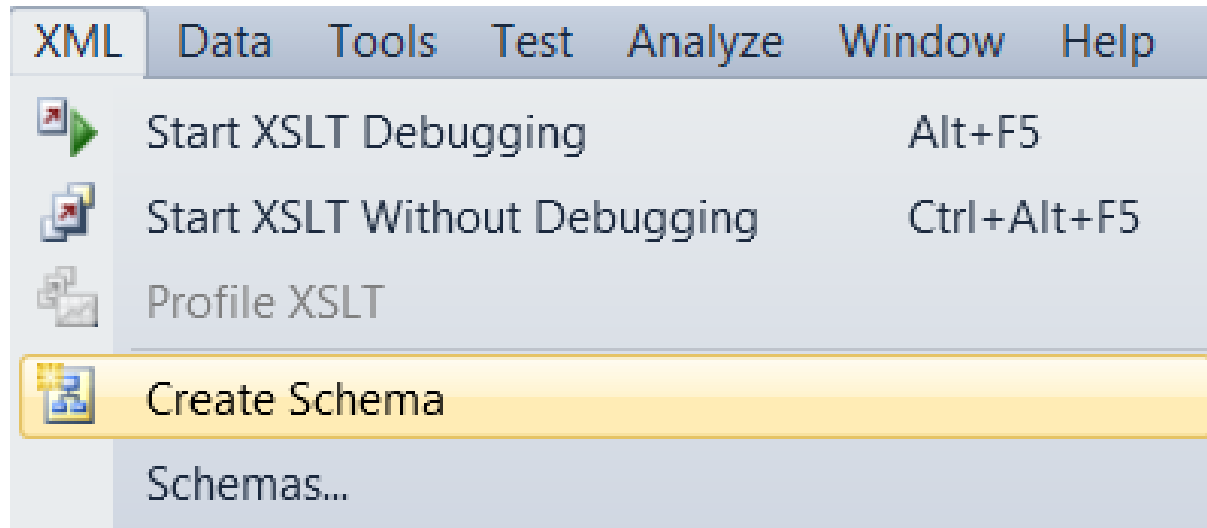
```
XMLFile1.xml x
<?xml version="1.0" encoding="utf-8"?>
```



```
<?xml version="1.0" encoding="utf-8"?>
<novels>
  <novel>
    <title>Ustam ve Ben</title>
    <author>
      <name>Elif Şafak</name>
      <nationality>Turkish</nationality>
    </author>
    <rating>Best Seller</rating>
  </novel>
  <novel>
    <title>Beyoğlu'nun En Güzel Abisi</title>
    <author>
      <name>Ahmet Ümit</name>
      <nationality>Turkish</nationality>
    </author>
    <rating>Best Seller</rating>
  </novel>
  <novel>
    <title>1984</title>
    <author>
      <name>George Orwell</name>
      <nationality>English</nationality>
    </author>
    <rating>Cult novel</rating>
  </novel>
</novels>
```

Uygulama - XML

XML şemasının yaratılması

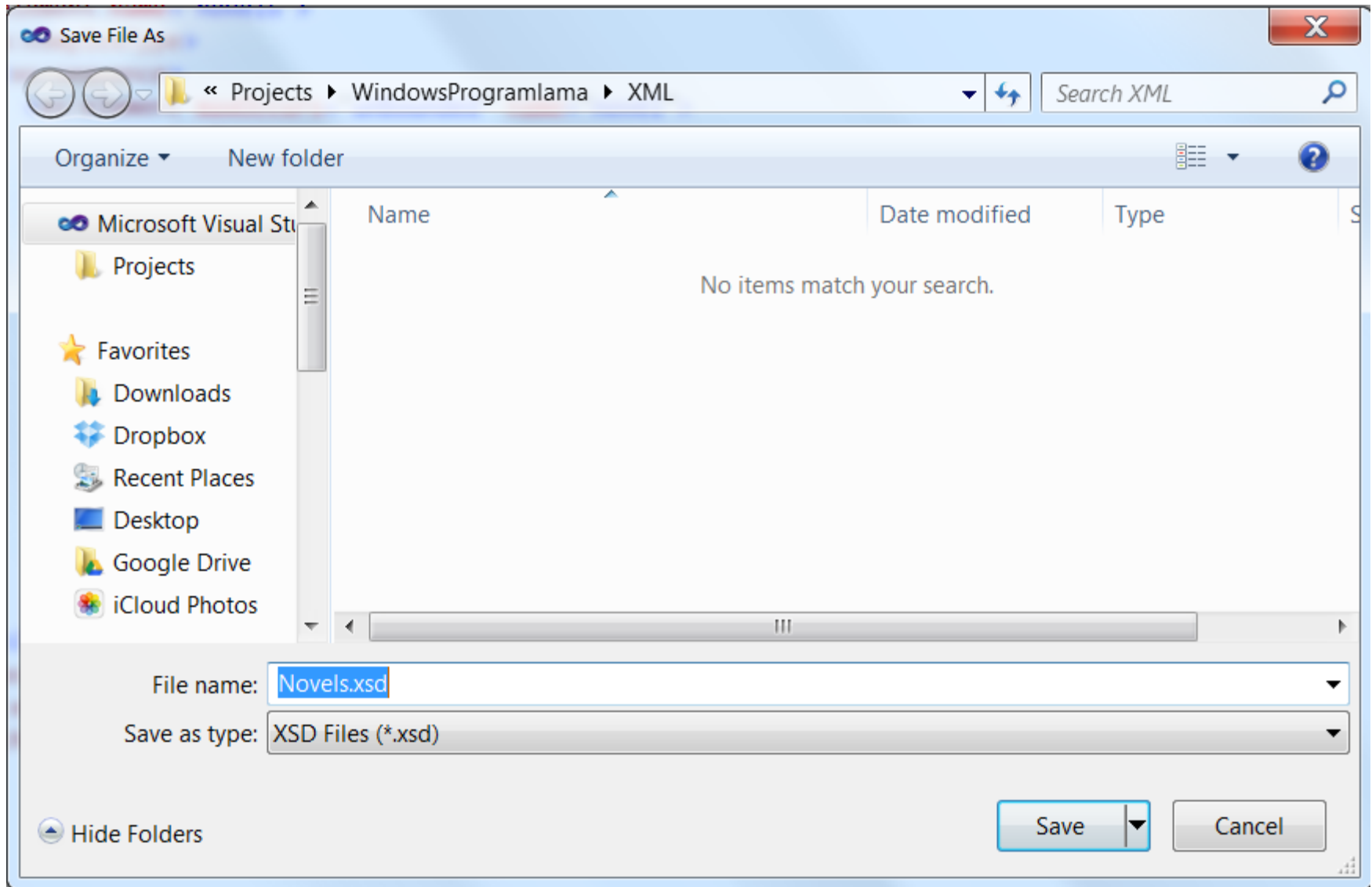


Uygulama - XML

Novels.xsd* X Novels.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="novels">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="novel">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string" />
              <xs:element name="author">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string" />
                    <xs:element name="nationality" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="rating" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Uygulama - XML

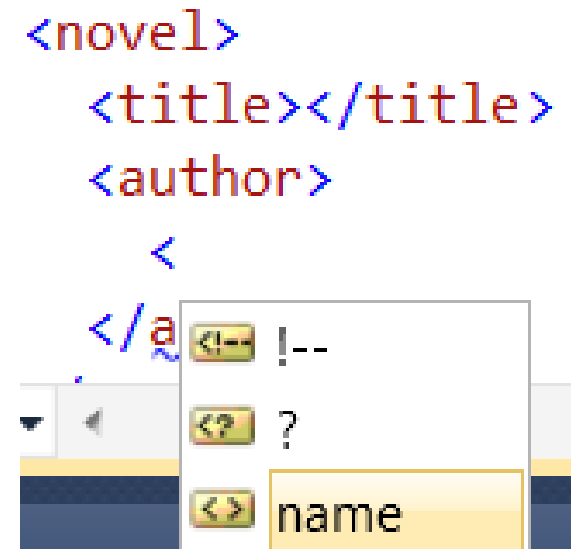


Uygulama - XML

Novels.xml dosyasına yeni bir roman eklenmesi:

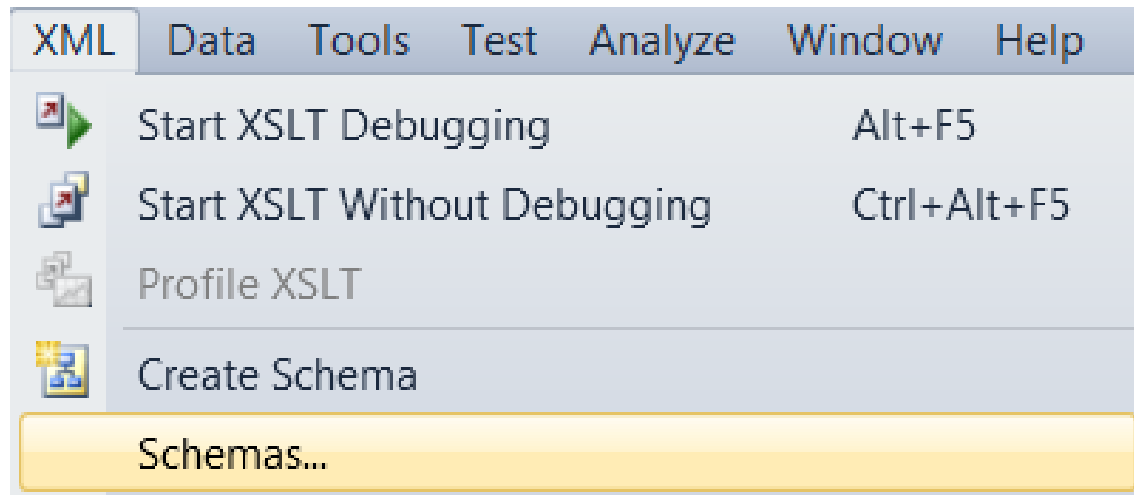
- Visual Studio XSD şemasını XML dosyasına bağlaması gerektiğini bildiğinden;
 - IntelliSense etiketleri gösterecektir.

```
<novel>
  <title>Lincoln</title>
  <author>
    <name>Gore Vidal</name>
    <nationality>American</nationality>
  </author>
  <rating>American History</rating>
</novel>
```



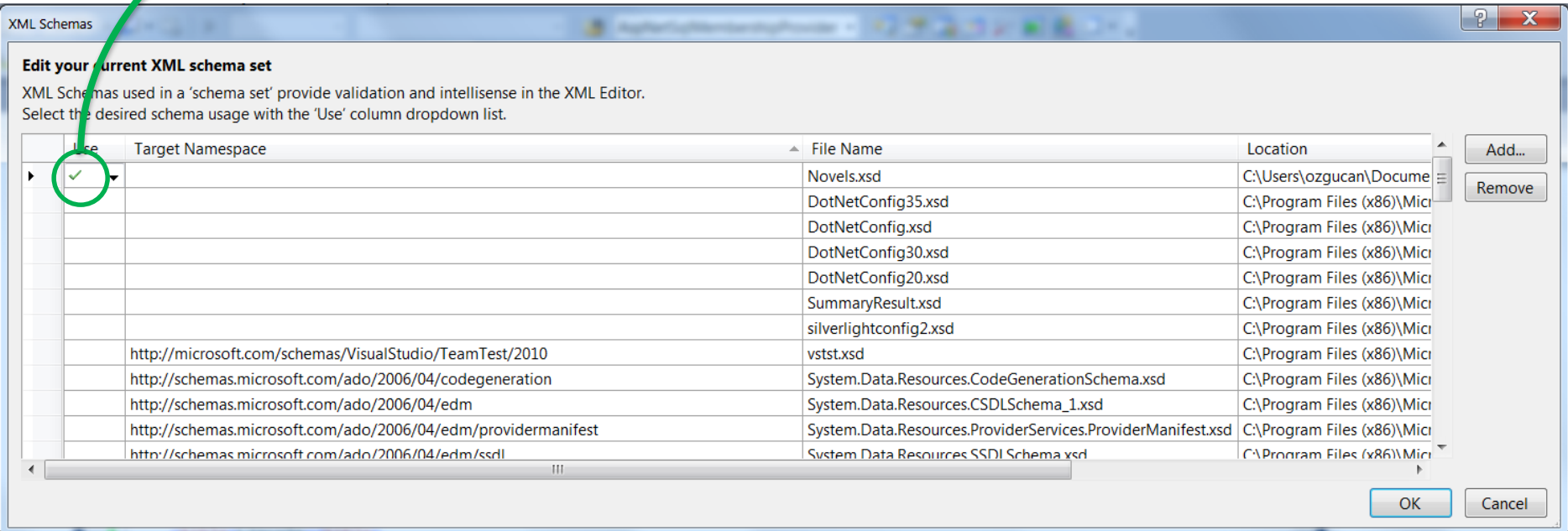
Uygulama - XML

- XML dosyası ve şema arasındaki bu bağlantı birden fazla şema ile de gerçekleştirilebilir.



Uygulama - XML

Mevcut XML dosyasının
hangi (Novels.xsd) şemayı
kullandığını belirtmektedir.



Sık kullanılan şemalar

C:\Program Files (x86)\Microsoft Visual Studio 10.0\Xml\Schemas
dizini altına kopyalanabilir.

.NET uygulamalarında XML'in okunması, değiştirilmesi ve yazılması

UYGULAMALARDA XML KULLANIMI

XML Document Object Model (DOM)

- XML DOM, XML'e erişim ve XML'de değişiklikler yapılması için çeşitli sınıflardan oluşan bir kümedir.
- DOM'u oluşturan sınıflar **System.Xml** ad uzayı altında bulunmaktadır.

DOM Sınıfları

CLASS	DESCRIPTION
<code>XmlNode</code>	Represents a single node in a document tree. It is the base of many of the classes shown in this chapter. If this node represents the root of an XML document, you can navigate to any position in the document from it.
<code>XmlDocument</code>	Extends the <code>XmlNode</code> class, but is often the first object you use when using XML. That's because this class is used to load and save data from disk or elsewhere.
<code>XmlElement</code>	Represents a single element in the XML document. <code>XmlElement</code> is derived from <code>XmlLinkedNode</code> , which in turn is derived from <code>XmlNode</code> .
<code>XmlAttribute</code>	Represents a single attribute. Like the <code>XmlDocument</code> class, it is derived from the <code>XmlNode</code> class.
<code>XmlText</code>	Represents the text between a starting tag and a closing tag.
<code>XmlComment</code>	Represents a special kind of node that is not regarded as part of the document other than to provide information to the reader about parts of the document.
<code>XmlNodeList</code>	Represents a collection of nodes.

XmlDocument Sınıfı

- Uygulamanın ilk işlemi, XML'in diskten okunmasıdır.
- **XmlDocument**, dosyanın disk üzerindeki bellek temsilidir.
- **XmlDocument**'den dosyanın belleğe yüklenmesi istendiğinde, bu dosyadan **XML belgesinin kökü (root) elde edilir ve belge okunur.**

XmlDocument Sınıfı

```
using System.Xml;
```

```
.....
```

```
.....
```

```
XmlDocument document = new XmlDocument();
```

```
document.Load(@"C:\WindowsPrg\Lab\Books.xml");
```

- **XmlDocument** sınıfının yeni bir **instance**'ı yaratılır.
- **Books.xml** dosyası bu **instance**'a yüklenir.

XmlElement Sınıfı

- **XmlElement**, XML belgesinde ki tek bir öğeyi temsil eder.

```
XmlDocument document = new XmlDocument();  
document.Load(@"C:\WindowsPrg\Lab\Books.xml");  
XmlElement element = document.Element;
```

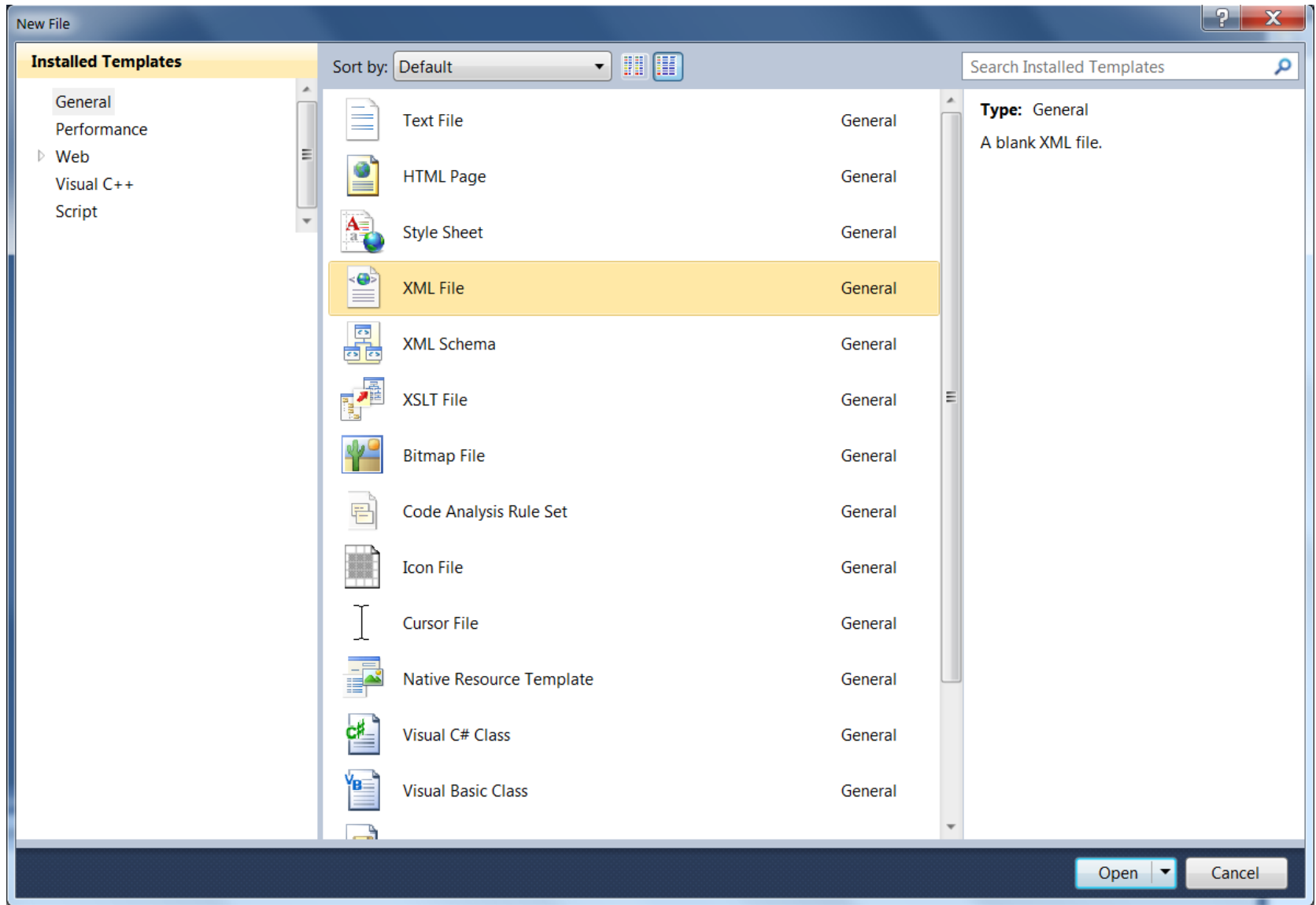
- **element** *instance*'ı **XmlDocument**'in kök (root) ögesini temsil eder.

XmlElement Özellikleri*

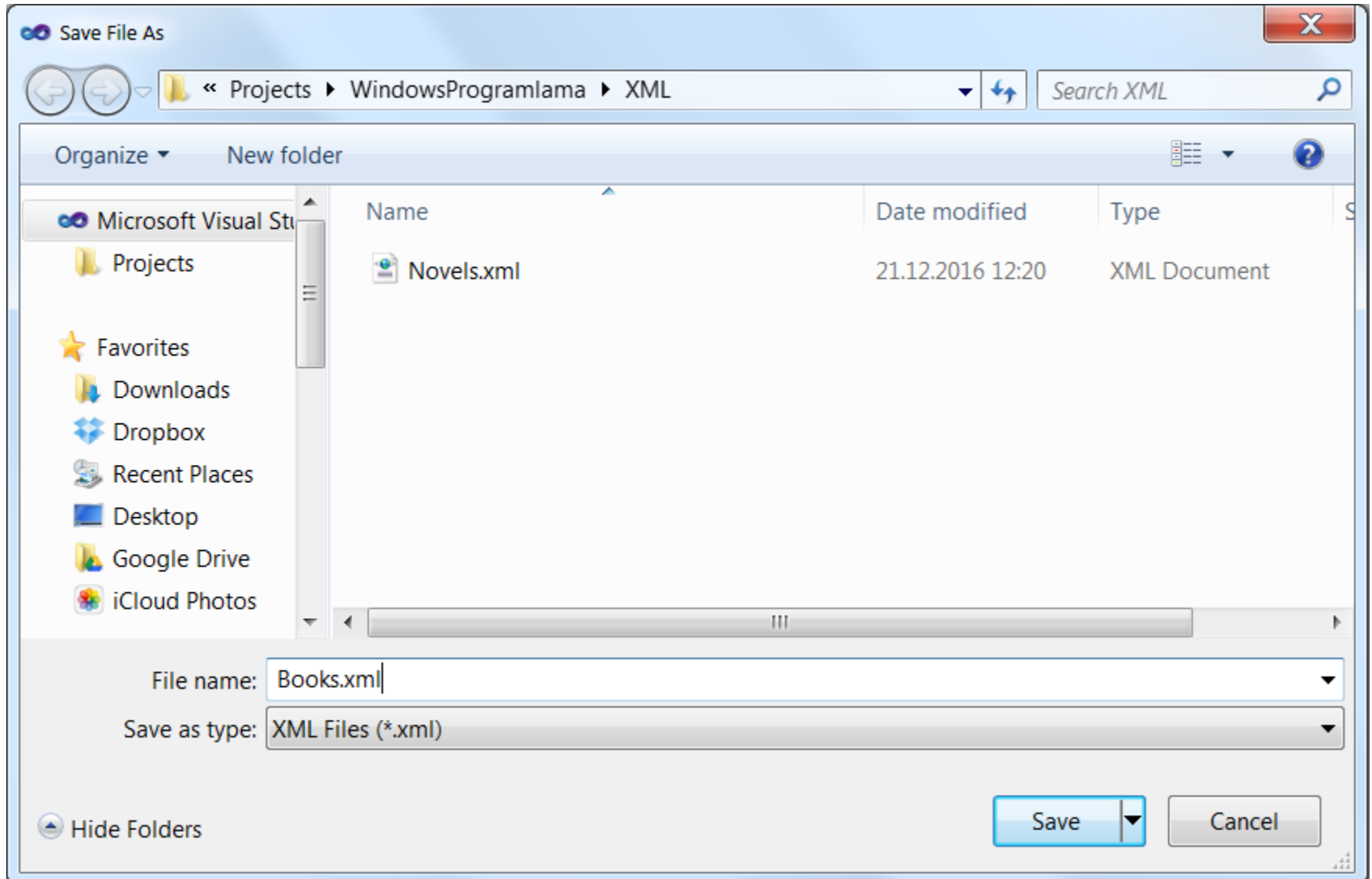
- **FirstChild**
 - Mevcut düğümün (node) alt (child) ögesini döndürür.
- **LastChild**
 - Mevcut düğümün son ögesini döndürür.
- **ParentNode**
 - Mevcut düğümün üst (parent) düğümünü döndürür.
- **NextSibling**
 - Aynı üst (parent) düğüme sahip olan diğer düğümü döndürür.
- **HasChildNodes**
 - Mevcut ögenin alt ögeleri olup olmadığını belirten bir değer döndürür.

* [http://msdn.microsoft.com/tr-tr/library/System.Xml.XmlElement_properties\(v=vs.110\).aspx](http://msdn.microsoft.com/tr-tr/library/System.Xml.XmlElement_properties(v=vs.110).aspx)

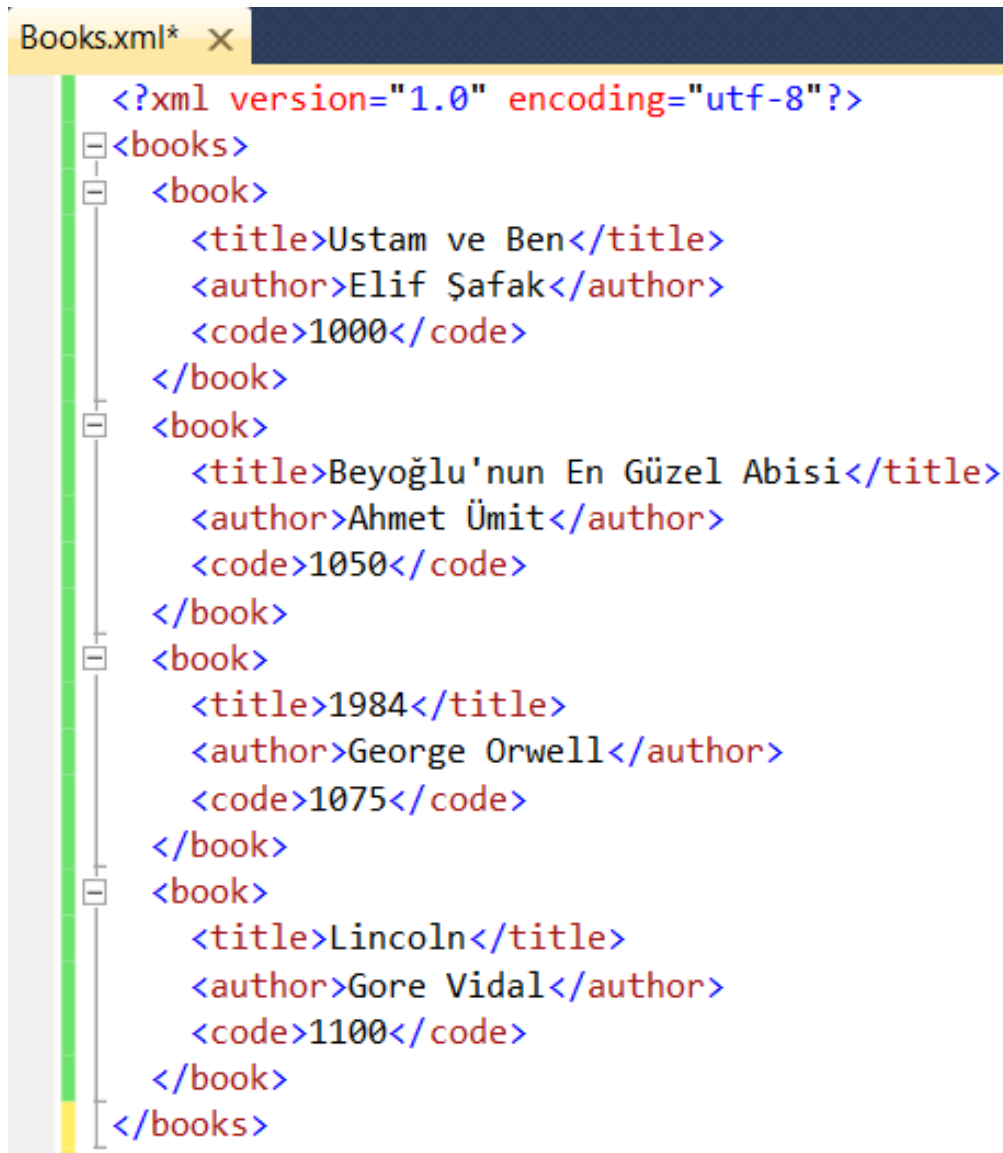
Uygulama - XML



Uygulama - XML

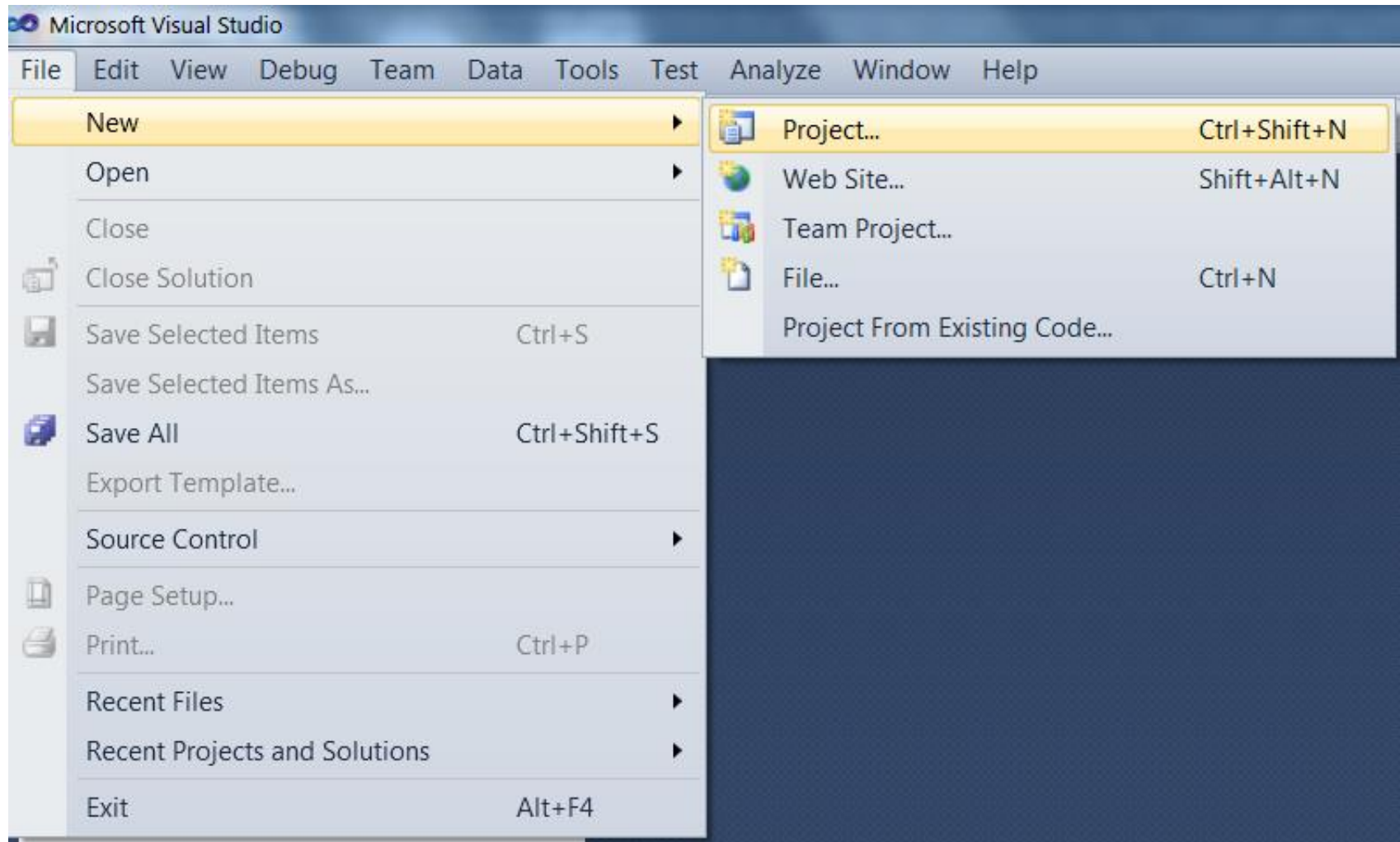


Uygulama - XML

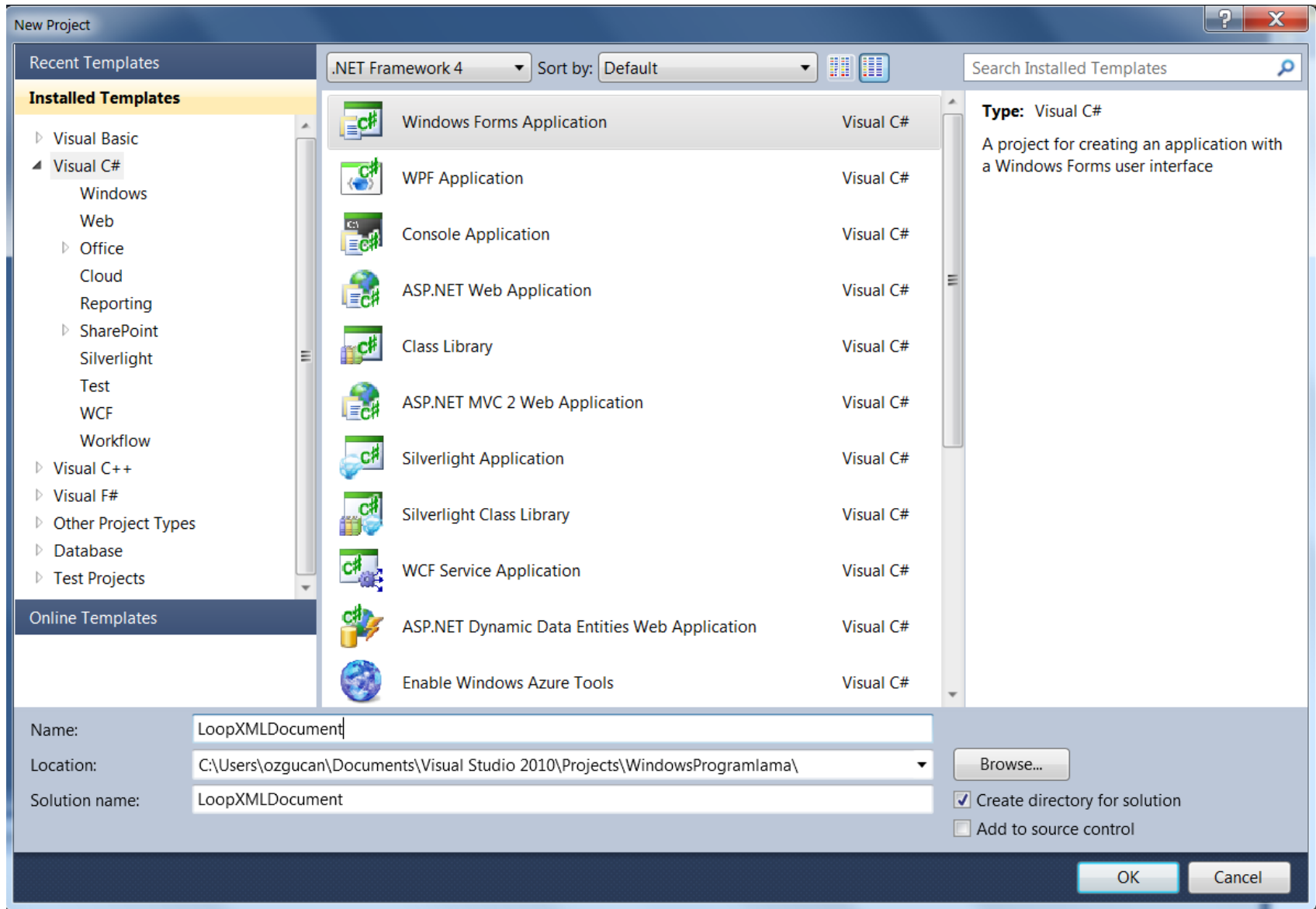


```
Books.xml* X
<?xml version="1.0" encoding="utf-8"?>
<books>
  <book>
    <title>Ustam ve Ben</title>
    <author>Elif Şafak</author>
    <code>1000</code>
  </book>
  <book>
    <title>Beyoğlu'nun En Güzel Abisi</title>
    <author>Ahmet Ümit</author>
    <code>1050</code>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <code>1075</code>
  </book>
  <book>
    <title>Lincoln</title>
    <author>Gore Vidal</author>
    <code>1100</code>
  </book>
</books>
```

Uygulama - XML



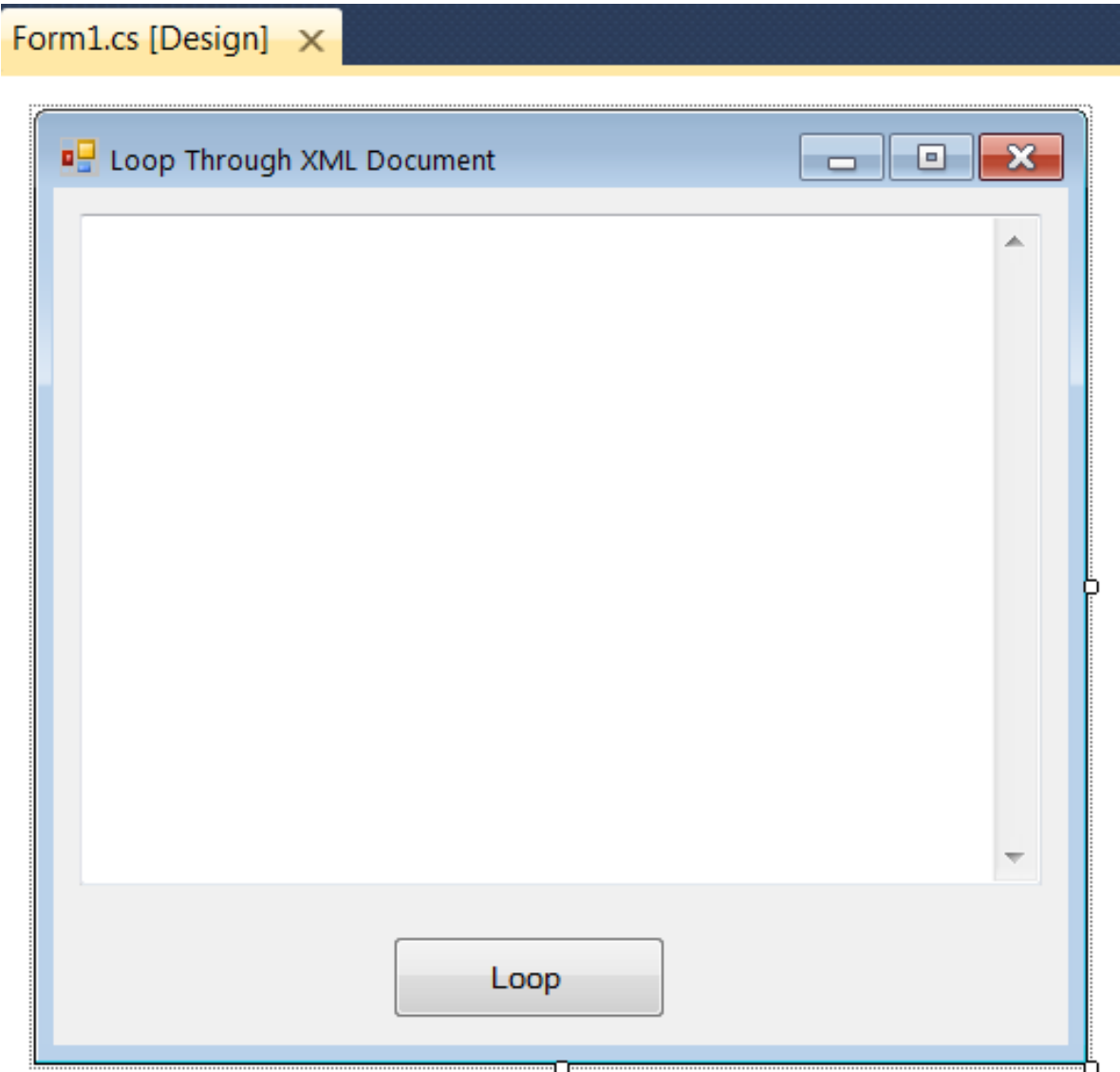
Uygulama - XML



Uygulama - XML

- **Form**
 - **Text = Loop Through XML Document**
- **TextBox**
 - **Name = resultTextBox**
 - **MultiLine = True**
 - **ScrollBars = Vertical**
- **Button**
 - **Name = loopButton**
 - **Text = Loop**

Uygulama - XML



Uygulama - XML

`using System.Xml;`  XML işlemleri için ilgili ad uzayının eklenmesi

```
private void loopButton_Click(object sender, EventArgs e)
{
    XmlDocument document = new XmlDocument();
    document.Load(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Books.xml");
    resultTextBox.Text = FormatText(document.DocumentElement as XmlNode, "", "");
}
```

 XML belgesinin kök (root) ögesi

```

private string FormatText(XmlNode node, string text, string indent)
{
    if (node is XmlText)
    {
        text += node.Value;
        return text;
    }

    if (string.IsNullOrEmpty(indent))
        indent = "";
    else
    {
        text += "\r\n" + indent;
    }

    if (node is XmlComment)
    {
        text += node.OuterXml;
        return text;
    }

    text += "<" + node.Name;
    if (node.Attributes.Count > 0)
    {
        AddAttributes(node, ref text);
    }
    if (node.HasChildNodes)
    {
        text += ">";
        foreach (XmlNode child in node.ChildNodes)
        {
            text = FormatText(child, text, indent + " ");
        }
        if (node.ChildNodes.Count == 1 &&
            (node.FirstChild is XmlText || node.FirstChild is XmlComment))
            text += "</" + node.Name + ">";
        else
            text += "\r\n" + indent + "</" + node.Name + ">";
    }
    else
        text += " />";
    return text;
}

```

Düğümün değeri

XmlElement

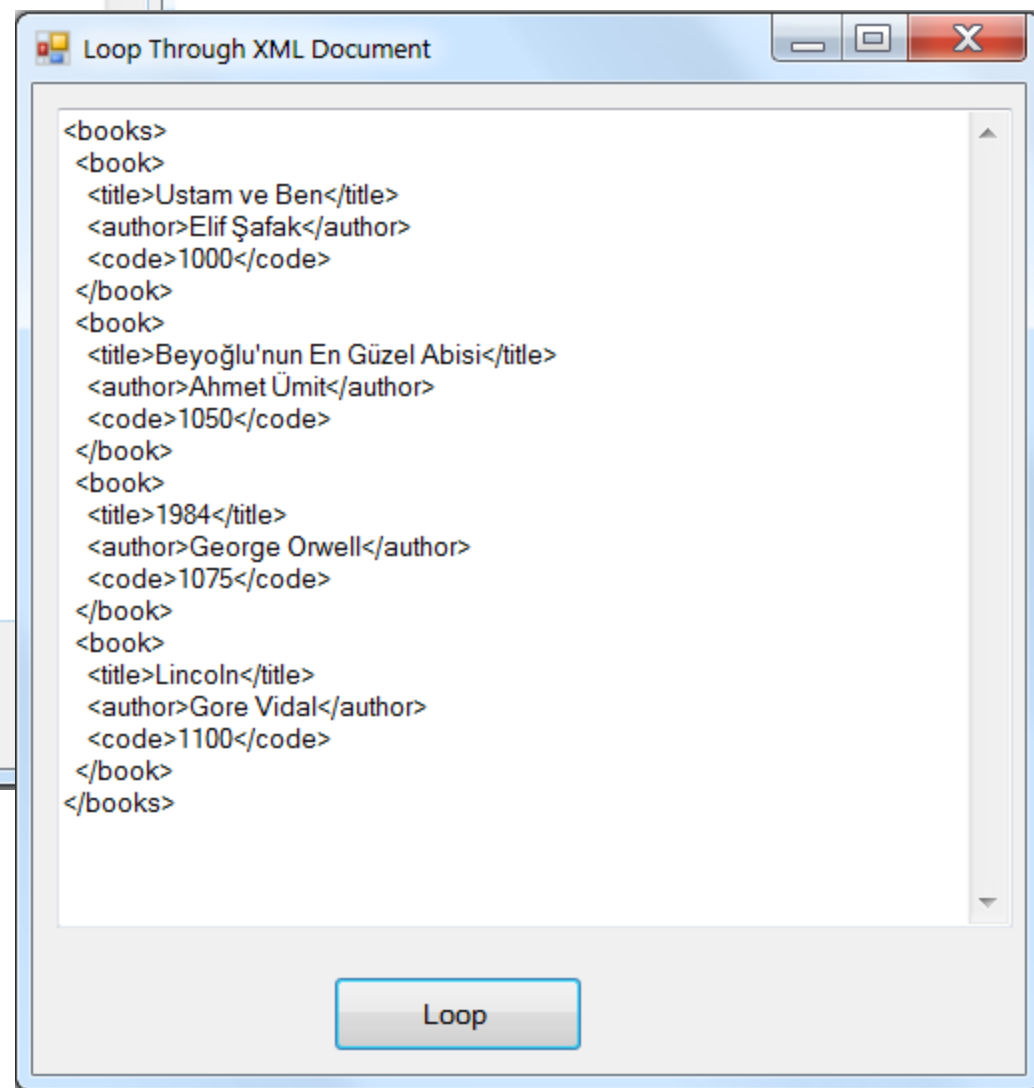
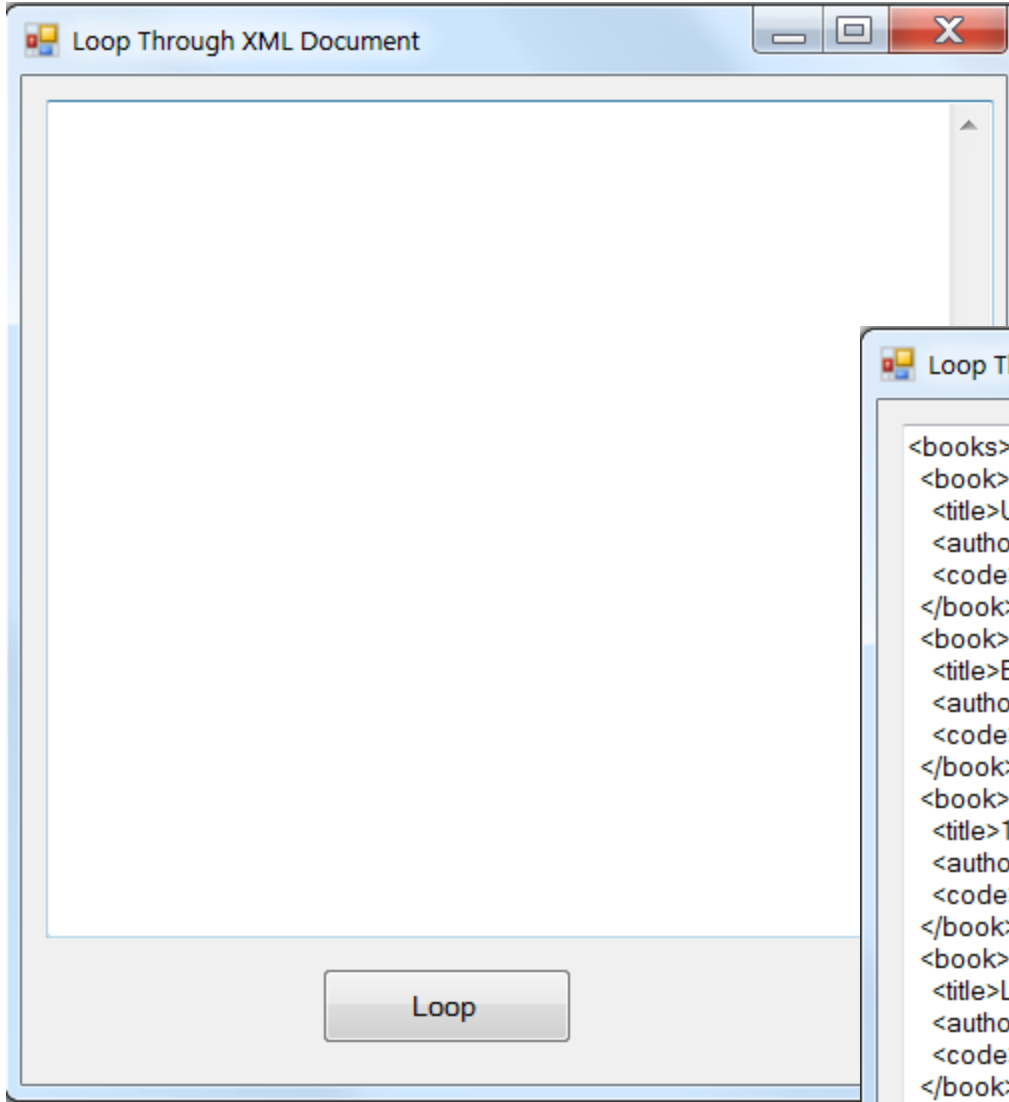
- Books
- Book
- Title
- Author
- Code

XmlText

Açma ve
kapama
etiketleri
arasında
kalan metin

Uygulama - XML

```
private void AddAttributes(XmlNode node, ref string text)
{
    foreach (XmlAttribute xa in node.Attributes)
    {
        text += " " + xa.Name + "=" + xa.Value + " ";
    }
}
```



Düğüm Eklenmesi Metotları

METHOD	DESCRIPTION
<code>AppendChild</code>	Appends a child node to a node of type <code>XmlNode</code> or a derived type. Remember that the node you append appears at the bottom of the list of children of the node on which the method is called. If you don't care about the order of the children, there's no problem; if you do care, remember to append the nodes in the correct sequence.
<code>InsertAfter</code>	Controls exactly where you want to insert the new node. The method takes two parameters — the first is the new node and the second is the node after which the new node should be inserted.
<code>InsertBefore</code>	Works exactly like <code>InsertAfter</code> , except that the new node is inserted before the node you supply as a reference.

Düğüm Eklenmesi Metotları

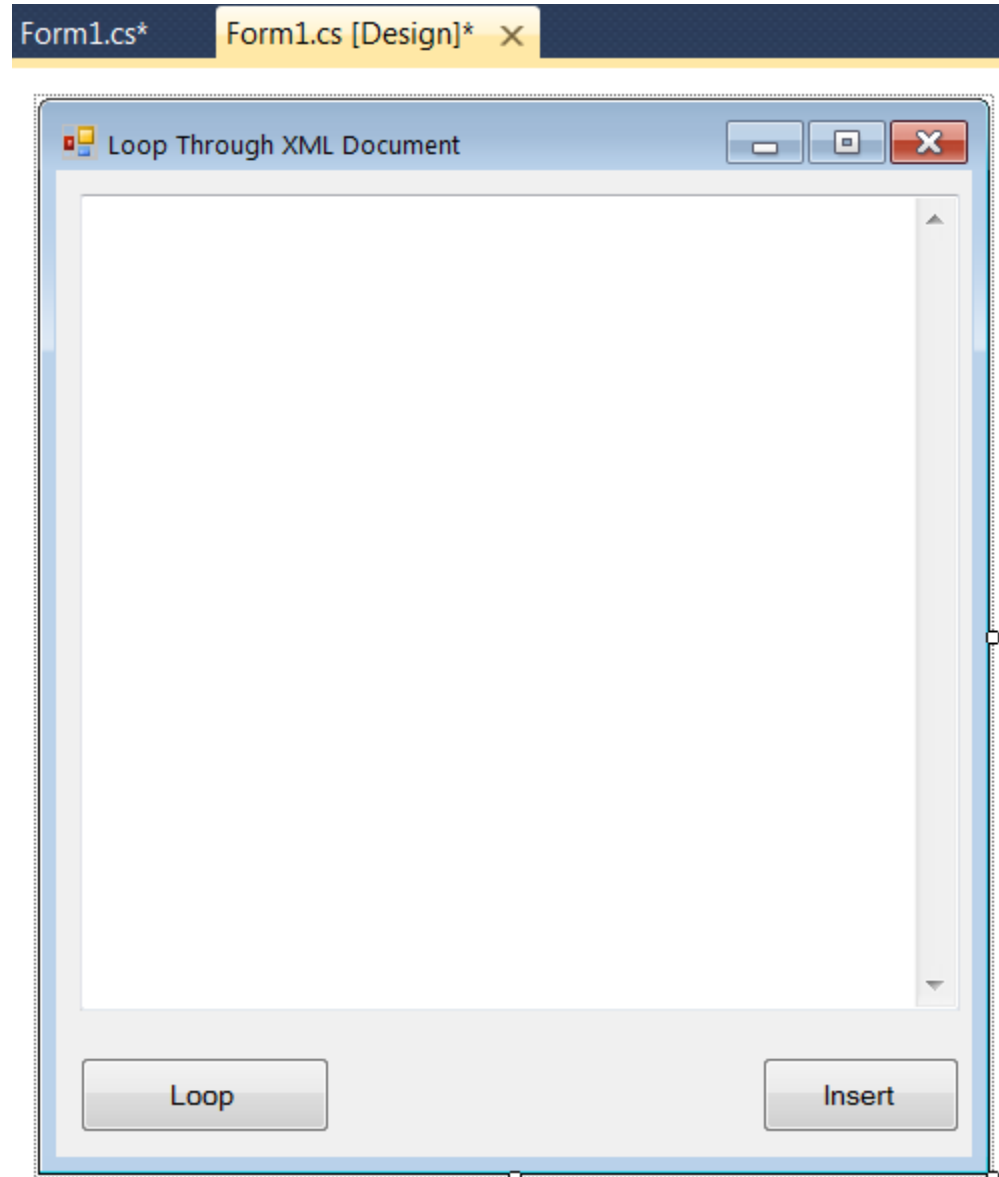
METHOD	DESCRIPTION
CreateNode	Creates any kind of node. There are three overloads of the method, two of which enable you to create nodes of the type found in the <code>XmlNodeType</code> enumeration and one that enables you to specify the type of node to use as a string. Unless you are quite sure about specifying a node type other than those in the enumeration, use the two overloads that use the enumeration. The method returns an instance of <code>XmlNode</code> that can then be cast to the appropriate type explicitly.
CreateElement	A version of <code>CreateNode</code> that creates only nodes of the <code>XmlElement</code> variety.
CreateAttribute	A version of <code>CreateNode</code> that creates only nodes of the <code>XmlAttribute</code> variety.
CreateTextNode	Creates — yes, you guessed it — nodes of the type <code>XmlTextNode</code>
CreateComment	This method is included here to highlight the diversity of node types that can be created. This method doesn't create a node that is actually part of the data represented by the XML document, but rather is a comment meant for any human eyes that might have to read the data. You can pick up comments when reading the document in your applications as well.

Uygulama - XML

Bir önceki uygulamaya
Button ekleyiniz.

Name = createButton

Text = Insert



Uygulama - XML

```
private void createButton_Click(object sender, EventArgs e)
{
    // Load the XML document.
    XmlDocument document = new XmlDocument();
    document.Load(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Books.xml");

    // Get the root element.
    XmlElement root = document.DocumentElement;

    // Create the new nodes.
    XmlElement newBook = document.CreateElement("book");
    XmlElement newTitle = document.CreateElement("title");
    XmlElement newAuthor = document.CreateElement("author");
    XmlElement newCode = document.CreateElement("code");
    XmlText title = document.CreateTextNode("Benim Adım Kırmızı");
    XmlText author = document.CreateTextNode("Orhan Pamuk");
    XmlText code = document.CreateTextNode("1200");
    XmlComment comment = document.CreateComment("The book won the International IMPAC Dublin Literary Award in 2003.");

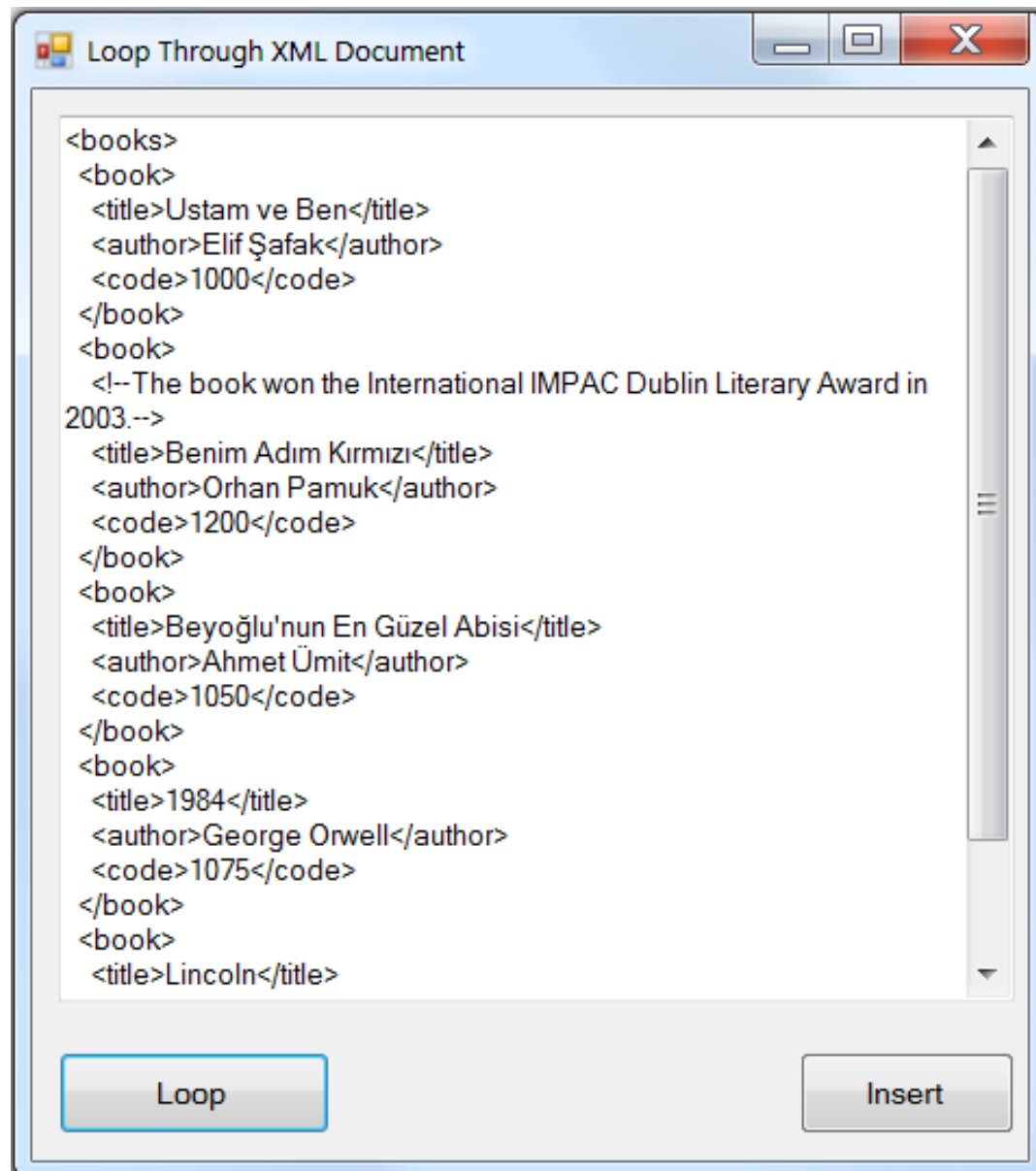
    // Insert the elements.
    newBook.AppendChild(comment);
    newBook.AppendChild(newTitle);
    newBook.AppendChild(newAuthor);
    newBook.AppendChild(newCode);
    newTitle.AppendChild(title);
    newAuthor.AppendChild(author);
    newCode.AppendChild(code);
    root.InsertAfter(newBook, root.FirstChild);

    document.Save(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Books.xml");
}
```

Düğümlerin yaratılması

Düğümler yaratıldıktan
sonra XML ağacına
eklenmesi

Uygulama - XML



Düğümlerin Silinmesi

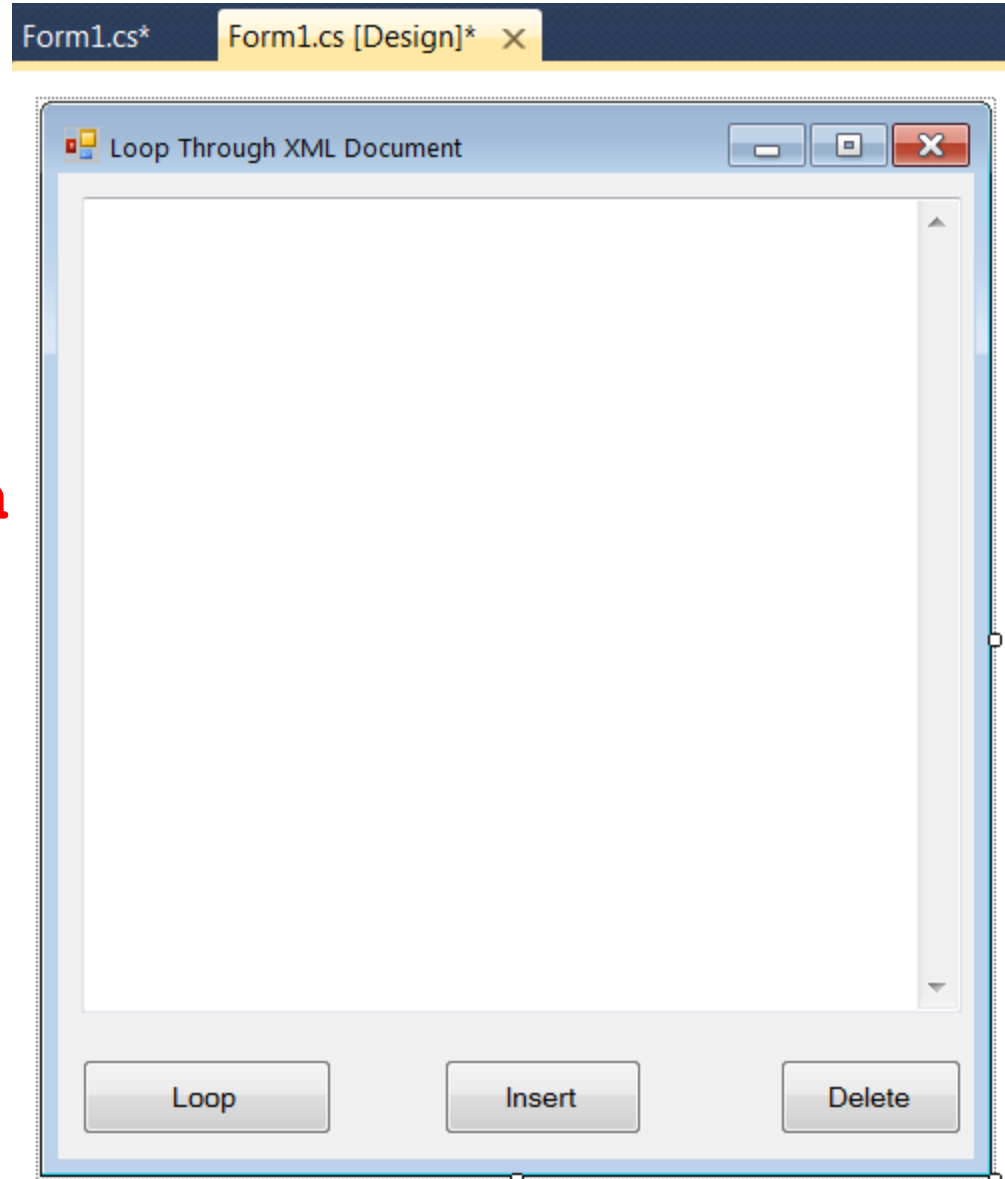
METHOD	DESCRIPTION
<code>RemoveAll</code>	Removes all child nodes in the node on which it is called. What is slightly less obvious is that it also removes all attributes on the node because they are regarded as child nodes as well.
<code>RemoveChild</code>	Removes a single child in the node on which it is called. The method returns the node that has been removed from the document, but you can reinsert it if you change your mind.

Uygulama - XML

Bir önceki uygulamaya
Button ekleyiniz.

Name = deleteButton

Text = Delete



Uygulama - XML

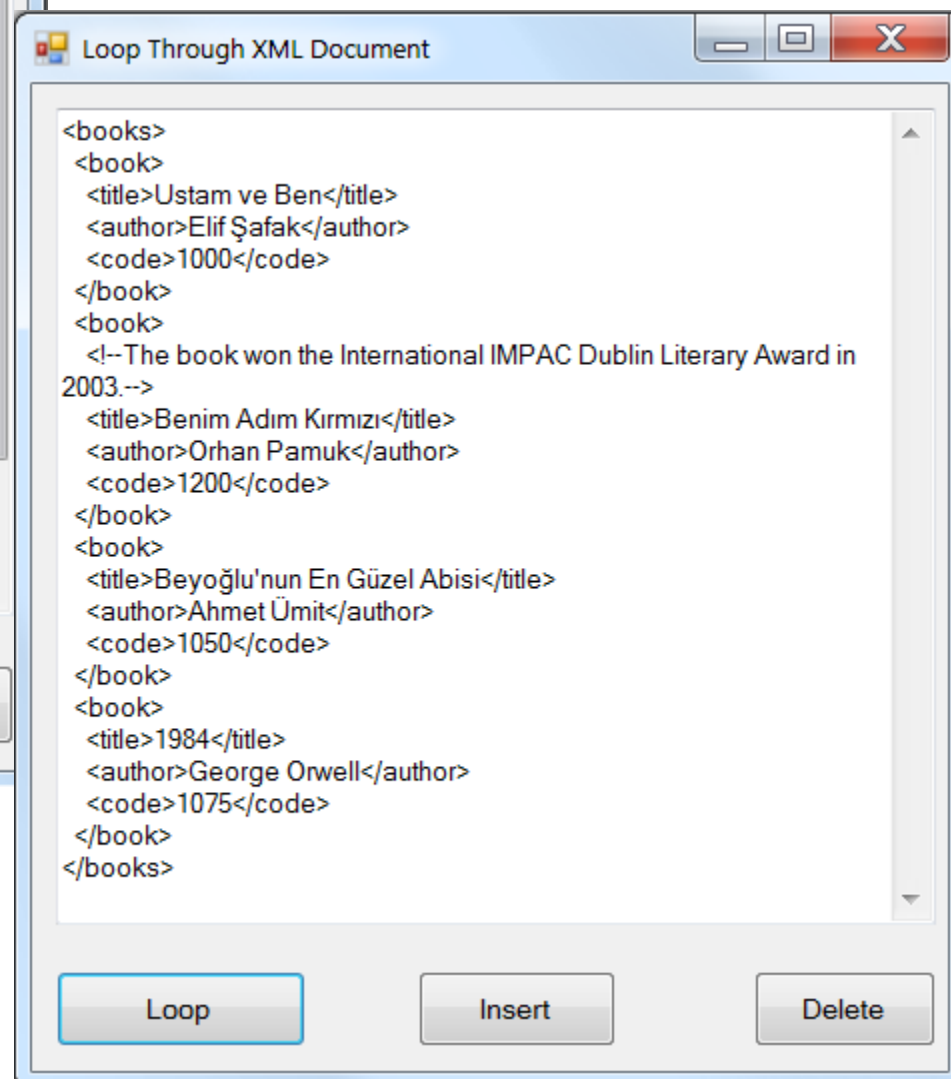
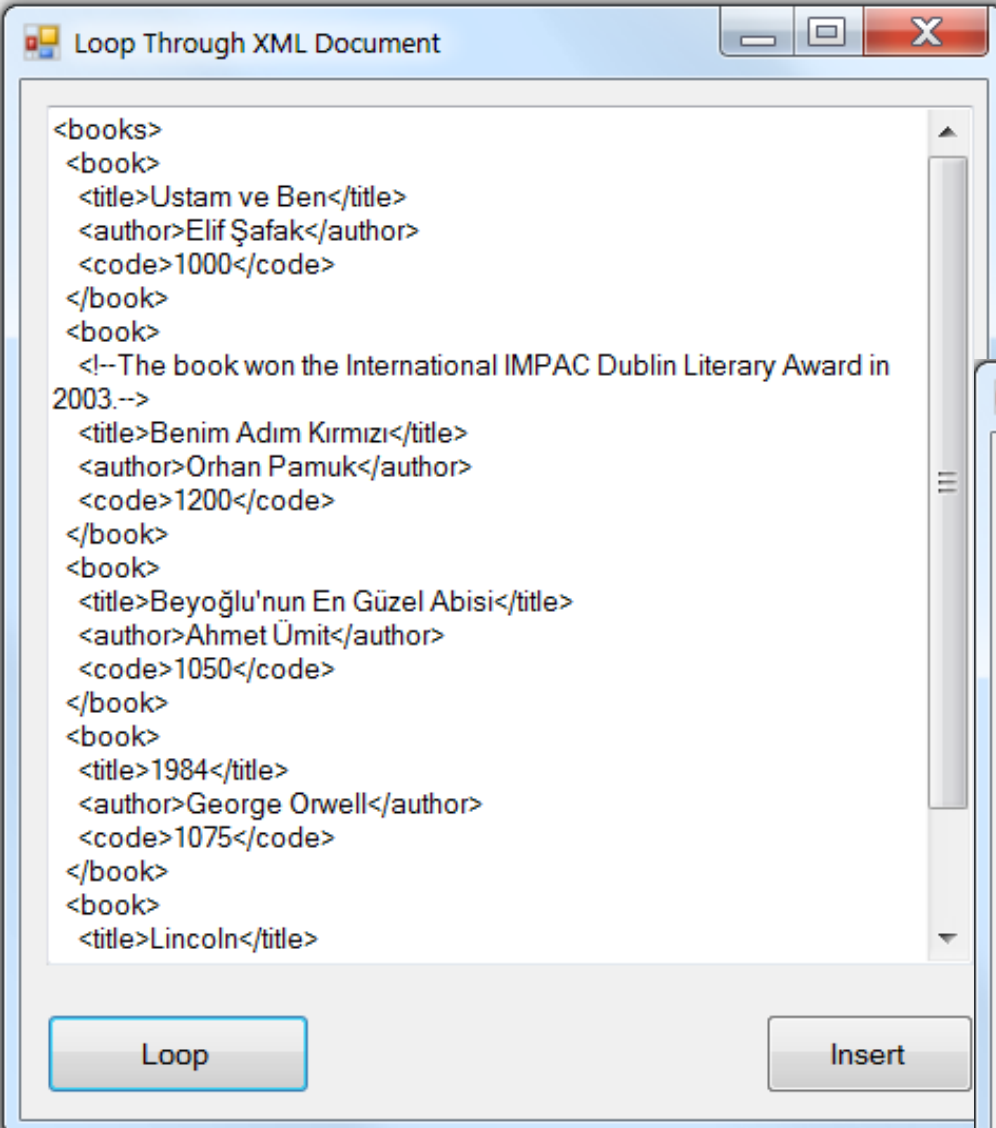
```
private void deleteButton_Click(object sender, EventArgs e)
{
    // Load the XML document.
    XmlDocument document = new XmlDocument();
    document.Load(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Books.xml");

    // Get the root element.
    XmlElement root = document.DocumentElement;

    // Find the node. root is the <books> tag, so its last child which will be the last <book> node.
    if (root.HasChildNodes)
    {
        XmlNode book = root.LastChild;

        // Delete the child.
        root.RemoveChild(book);

        // Save the document back to disk.
        document.Save(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Books.xml");
    }
}
```



Düğümlerin Seçilmesi

Bütün ağacı dolaşmadan düğümünün seçilmesi

- **XmlNode** sınıfının iki metodu:

METHOD	DESCRIPTION
SelectSingleNode	Selects a single node. If you create a query that fetches more than one node, only the first node will be returned.
SelectNodes	Returns a node collection in the form of an XmlNodeList class

Xpath*

- XML belgeleri için bir sorgulama dilidir.
 - İlişkisel veritabanlarındaki SQL gibidir.
- **SelectSingleNode** ve **SelectNodes** metotlarını kullanır.

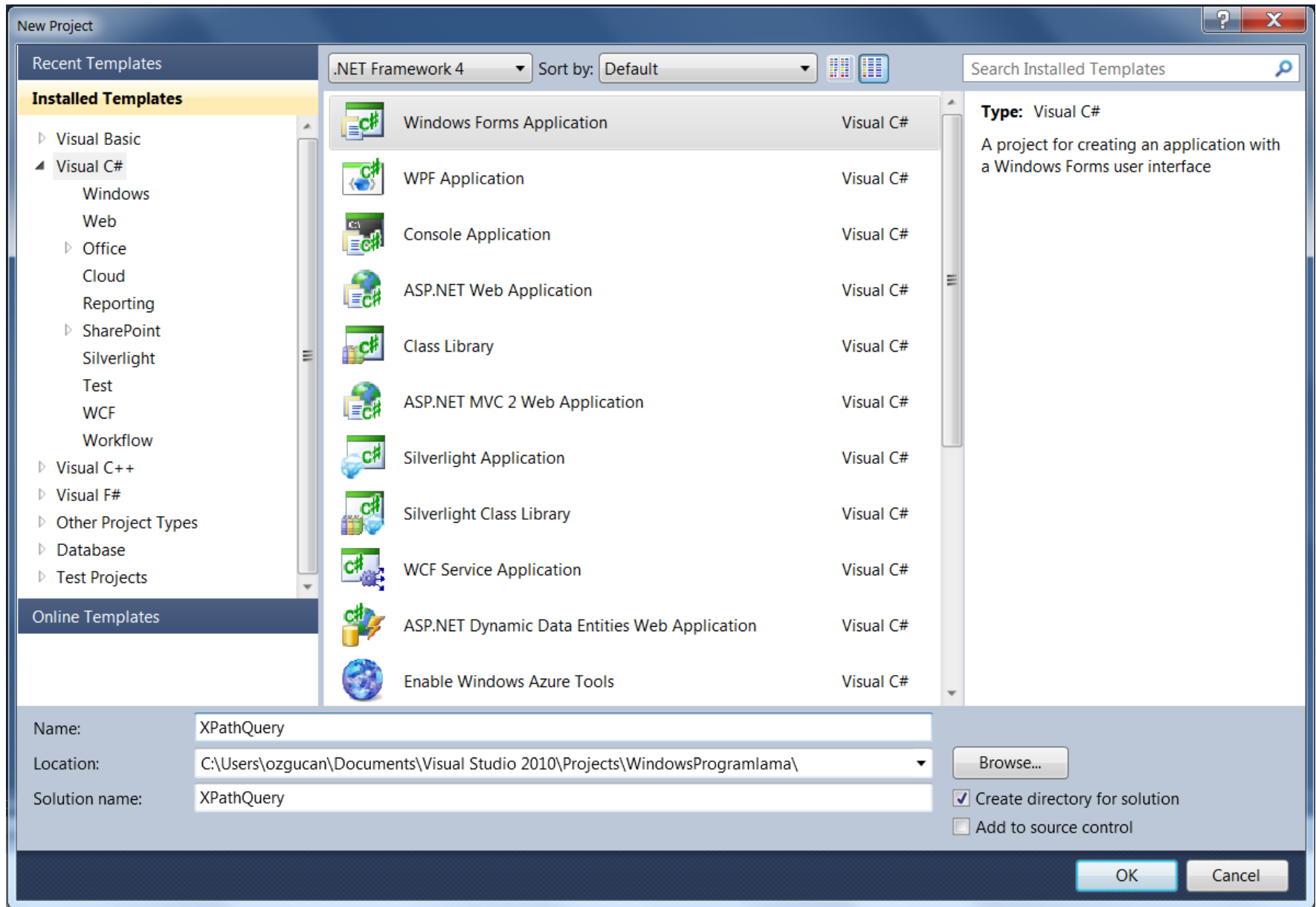
METHOD	DESCRIPTION
SelectSingleNode	Selects a single node. If you create a query that fetches more than one node, only the first node will be returned.
SelectNodes	Returns a node collection in the form of an XmlNodeList class

* <http://www.w3.org/TR/xpath/>
<http://www.w3schools.com/xpath/>

Xpath – En Sık Kullanılan İşlemler

PURPOSE	XPATH QUERY EXAMPLE
Select the current node.	.
Select the parent of the current node.	..
Select all child nodes of the current node.	*
Select all child nodes with a specific name — In this case, title.	title
Select an attribute of the current node.	@Type
Select all attributes of the current node.	@*
Select a child node by Index — In this case, the second element node.	element[2]
Select all the text nodes of the current node.	text()
Select one or more grandchildren of the current node.	element/text()
Select all nodes in the document with a particular name — In this case, all mass nodes.	//mass
Select all nodes in the document with a particular name and a particular parent name — In this case, the parent name is element and the node name is name.	//element/name
Select a node where a value criterion is met — In this case, the element for which the name of the element is Hydrogen.	//element[name='Hydrogen']
Select a node where an attribute value criterion is met — In this case, the Type attribute = Noble Gas.	//element[@Type='Noble Gas']

Uygulama - XML



Uygulama - XML

- **Form**
 - Text = Xpath Query Example
- **Label**
 - Name = queryLabel
 - Text = Query:
- **TextBox**
 - Name = queryTextBox
 - Multiline = True
- **Button**
 - Name = executeButton
 - Text = Execute
- **TextBox**
 - Name = resultTextBox
 - Multiline = True
 - ScrollBars = Vertical

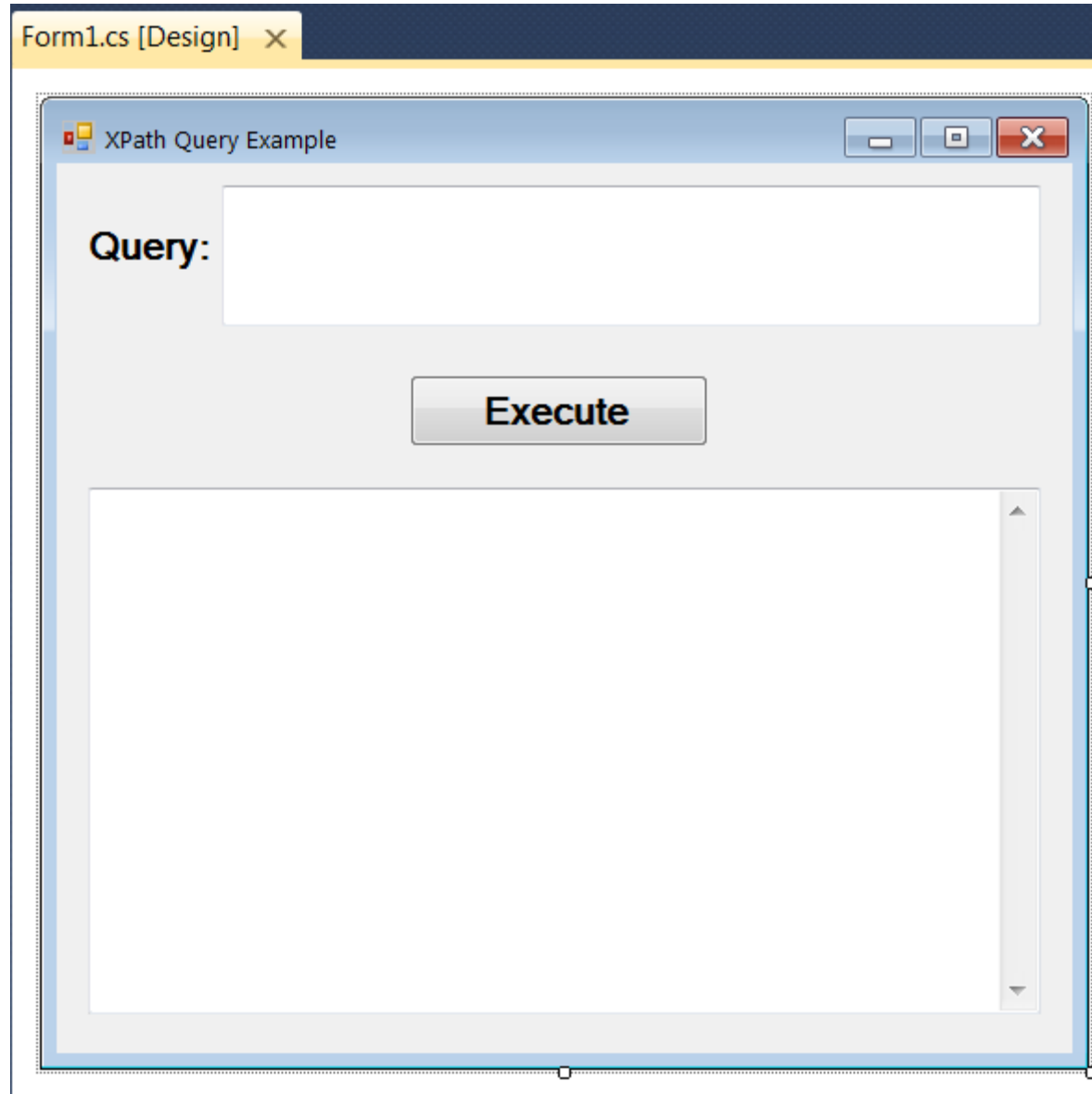
Uygulama - XML

Form1.cs [Design] X

XPath Query Example

Query:

Execute



The image shows a Windows Forms application in Design mode. The title bar indicates the file is 'Form1.cs [Design]'. The application window is titled 'XPath Query Example'. It contains a text box labeled 'Query:' for entering an XPath query. Below the text box is an 'Execute' button. At the bottom of the window is a large text area for displaying the results of the query. The application is currently in Design mode, as indicated by the 'X' icon in the title bar.

Uygulama - XML

`using System.Xml;`  XML işlemleri için ilgili ad uzayının eklenmesi

```
private XmlDocument document;
```

```
public Form1()
```

```
{
```

```
    InitializeComponent();
```

```
    document = new XmlDocument();
```

```
    document.Load(@"C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\XML\Elements.xml");
```

```
    Update(document.DocumentElement.SelectNodes("."));
```

```
}
```

Uygulama - XML

Sorgu sonuçlarının `resultTextBox`'da görüntülenmesi:

```
private void Update(XmlNodeList nodes)
{
    if (nodes == null || nodes.Count == 0)
    {
        resultTextBox.Text = "The query yielded no results";
        return;
    }
    string text = "";
    foreach (XmlNode node in nodes)
    {
        text = FormatText(node, text, "") + "\r\n";
    }
    resultTextBox.Text = text;
}
```

```

private string FormatText(XmlNode node, string text, string indent)
{
    if (node is XmlText)
    {
        text += node.Value;
        return text;
    }

    if (string.IsNullOrEmpty(indent))
        indent = "";
    else
    {
        text += "\r\n" + indent;
    }

    if (node is XmlComment)
    {
        text += node.OuterXml;
        return text;
    }

    text += "<" + node.Name;
    if (node.Attributes.Count > 0)
    {
        AddAttributes(node, ref text);
    }
    if (node.HasChildNodes)
    {
        text += ">";
        foreach (XmlNode child in node.ChildNodes)
        {
            text = FormatText(child, text, indent + " ");
        }
        if (node.ChildNodes.Count == 1 &&
            (node.FirstChild is XmlText || node.FirstChild is XmlComment))
            text += "</" + node.Name + ">";
        else
            text += "\r\n" + indent + "</" + node.Name + ">";
    }
    else
        text += " />";
    return text;
}

```

Uygulama - XML

```
private void AddAttributes(XmlNode node, ref string text)
{
    foreach (XmlAttribute xa in node.Attributes)
    {
        text += " " + xa.Name + "='" + xa.Value + "'";
    }
}
```

```
private void executeButton_Click(object sender, EventArgs e)
{
    try
    {
        XmlNodeList nodes = document.DocumentElement.SelectNodes(queryTextBox.Text);
        Update(nodes);
    }
    catch (Exception err)
    {
        resultTextBox.Text = err.Message;
    }
}
```

*Kullanıcının girdiği
sorgunun çalıştırılması*

XPath Query Example

Query:

Execute

```
<elements>
<!--First Non-Metal-->
<element Type='Non-Metal'>
  <name>Hydrogen</name>
  <symbol>H</symbol>
  <number>1</number>
  <specification>
    <mass>1.007825</mass>
    <density>0.0899 g/cm3</density>
  </specification>
</element>
<!--First Noble Gas-->
<element Type='Noble Gas'>
  <name>Helium</name>
  <symbol>He</symbol>
  <number>2</number>
  <specification>
```

XPath Query Example

Query:

Execute

```
<element Type='Non-Metal'>
  <name>Hydrogen</name>
  <symbol>H</symbol>
  <number>1</number>
  <specification>
    <mass>1.007825</mass>
    <density>0.0899 g/cm3</density>
  </specification>
</element>
```


Query:

`element[@Type='Noble Gas']`

Execute

```
<element Type='Noble Gas'>
  <name>Helium</name>
  <symbol>He</symbol>
  <number>2</number>
  <specification>
    <mass>4.002602</mass>
    <density>0.1785 g/cm3</density>
  </specification>
</element>
<element Type='Noble Gas'>
  <name>Neon</name>
  <symbol>Ne</symbol>
  <number>10</number>
  <specification>
    <mass>20.1797</mass>
    <density>0.901 g/cm3</density>
  </specification>
```

Query:

`element[/*]`

Execute

```
<element Type='Non-Metal'>
  <name>Hydrogen</name>
  <symbol>H</symbol>
  <number>1</number>
  <specification>
    <mass>1.007825</mass>
    <density>0.0899 g/cm3</density>
  </specification>
</element>
<element Type='Noble Gas'>
  <name>Helium</name>
  <symbol>He</symbol>
  <number>2</number>
  <specification>
    <mass>4.002602</mass>
    <density>0.1785 g/cm3</density>
  </specification>
```