XML Processing

Parsing XML

XDocument and LINQ

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

cc BY NC SA

Technical Trainers

Software University

Table of Contents

Have a Question?

SoftUni Foundation

sli.do

#csharp-db

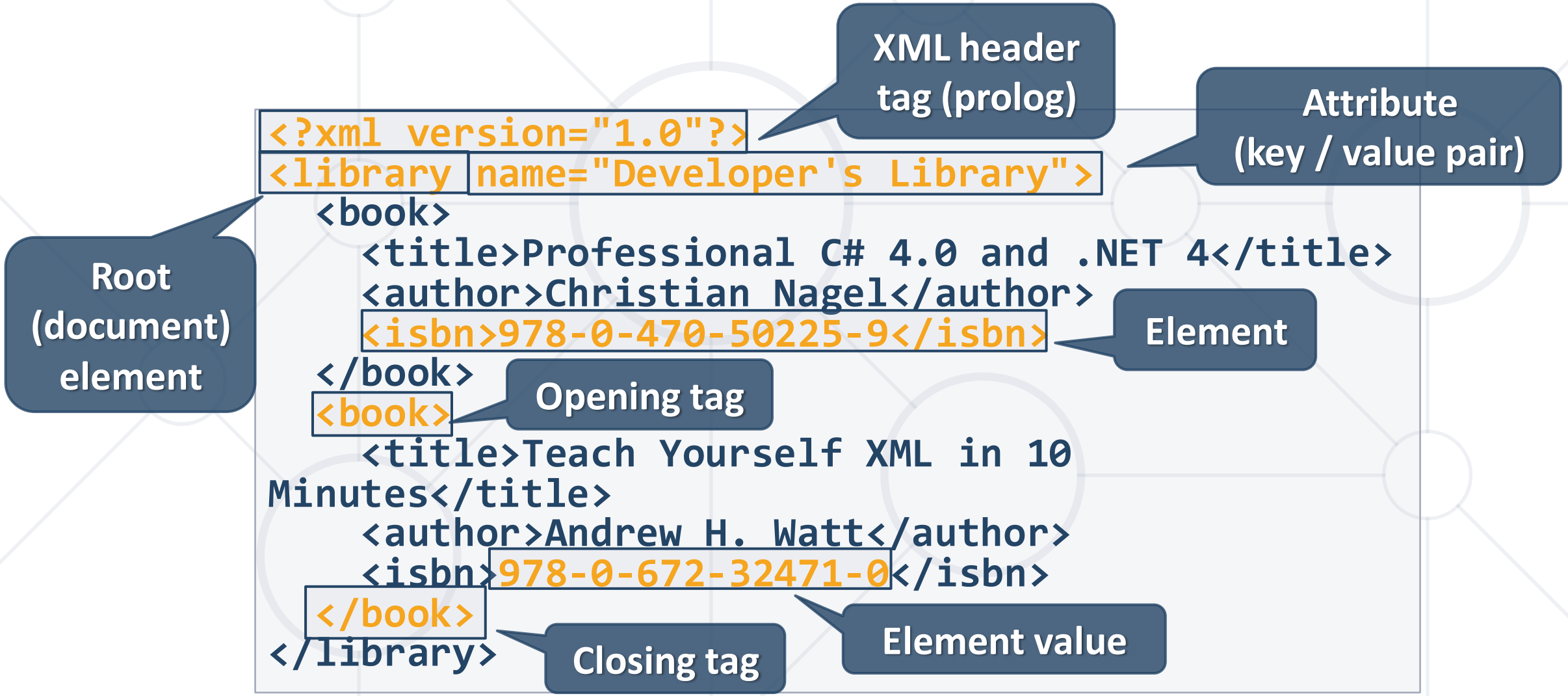
What is XML?

Format Description and Application

What is XML?

- EXtensible Markup Language
  - **Universal notation** (data format / language) for describing structured data using text with tags
  - Designed to **store** and **transport** data
  - The data is stored together with the **meta-data** about it

XML – Example


XML header tag (prolog)
Attribute (key / value pair)
Root (document) element
Element
Opening tag
Element value
Closing tag
<?xml version="1.0"?>
<library name="Developer's Library">
<book>
<title>Professional C# 4.0 and .NET 4</title>
<author>Christian Nagel</author>
<isbn>978-0-470-50225-9</isbn>
</book>
<book>
<title>Teach Yourself XML in 10 Minutes</title>
<author>Andrew H. Watt</author>
<isbn>978-0-672-32471-0</isbn>
</book>
</library>

XML Syntax

SoftUni Foundation

- Header – defines a **version** and character **encoding**

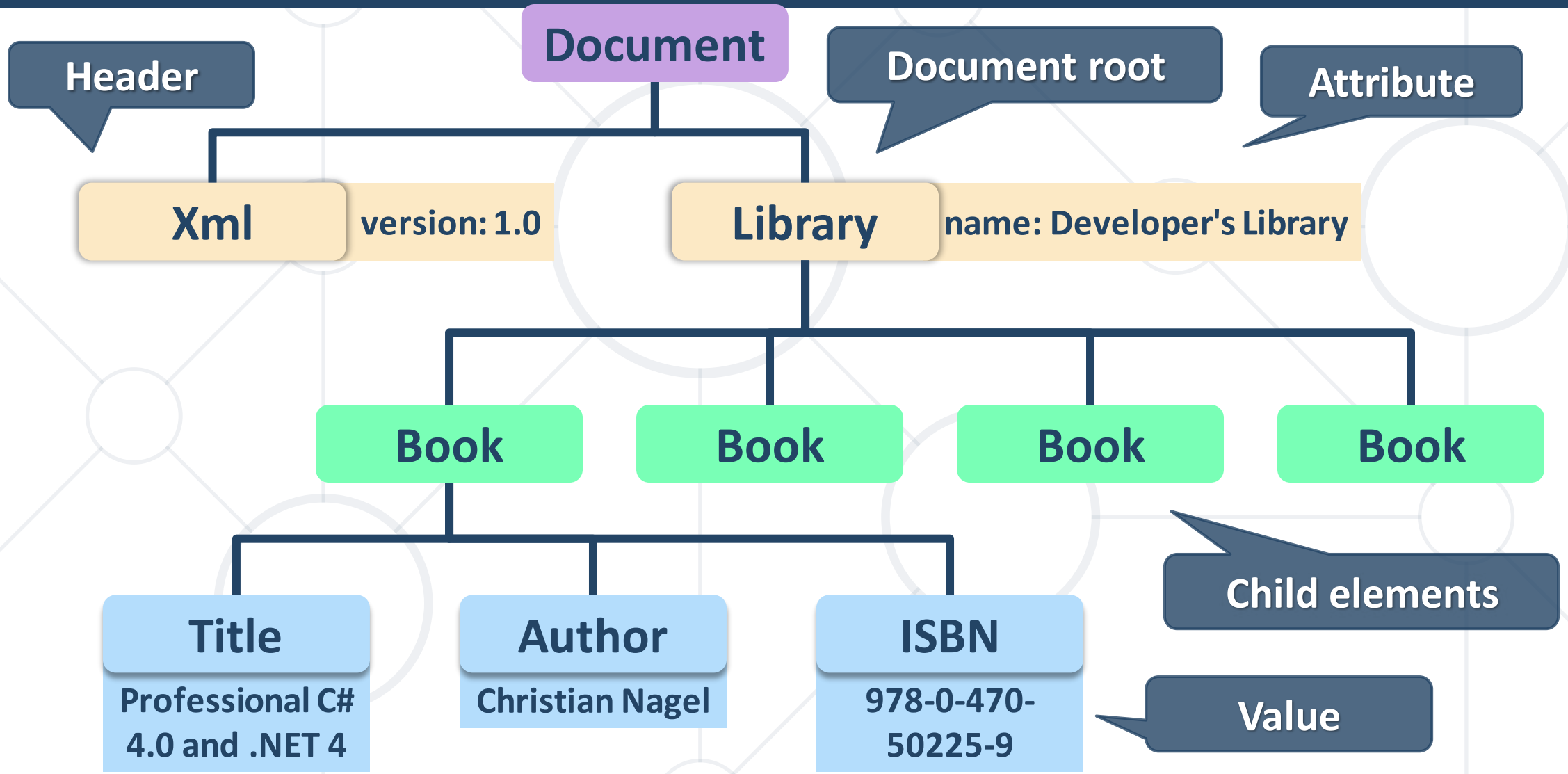
```
<?xml version="1.0" encoding="UTF-8"?>
```

- **Elements** – define the structure
- **Attributes** – element metadata
- **Values** – actual data, that can also be nested elements


```
<title lang="en">Professional C# 4.0 and .NET 4</title>
```

- Root element – required to **only** have **one**

XML - Structure



XML and HTML

- Similarities between XML and HTML
  - Both are **text based** notations
  - Both use **tags** and **attributes**
- Differences between XML and HTML
  - HTML describes documents, XML is a syntax for describing other languages (**meta-language**)
  - HTML describes the **layout** and the structure of information
  - XML requires the documents to be **well-formatted**

XML: Advantages

SoftUni Foundation

- Advantages of XML:
  - XML is **human-readable** (unlike binary formats)
  - Stores any kind of **structured data**
  - Data comes with self-describing **meta-data**
  - Full Unicode support
  - Custom XML-based languages can be designed for certain apps
  - **Parsers** available for virtually all languages and platforms

XML: Disadvantages

- Disadvantages of XML:
  - XML data is **bigger** (takes more space) than binary or JSON
    - More memory consumption, more network traffic, more hard-disk space, more resources, etc.
  - **Decreased performance**
    - CPU consumption: need of parsing / constructing the XML tags
- XML is **not** suitable for **all** kinds of **data**
  - E.g. binary data: graphics, images, videos, etc.

Parsing XML

Using XDocument and LINQ

LINQ to XML

- LINQ to XML
  - Use the power of **LINQ** to process XML data
  - Easily read, search, write, modify XML documents
- LINQ to XML classes:
  - **XDocument** – represents a LINQ-enabled XML document (containing prolog, root element, …)
  - **XElement** – main component holding information

Reading XML

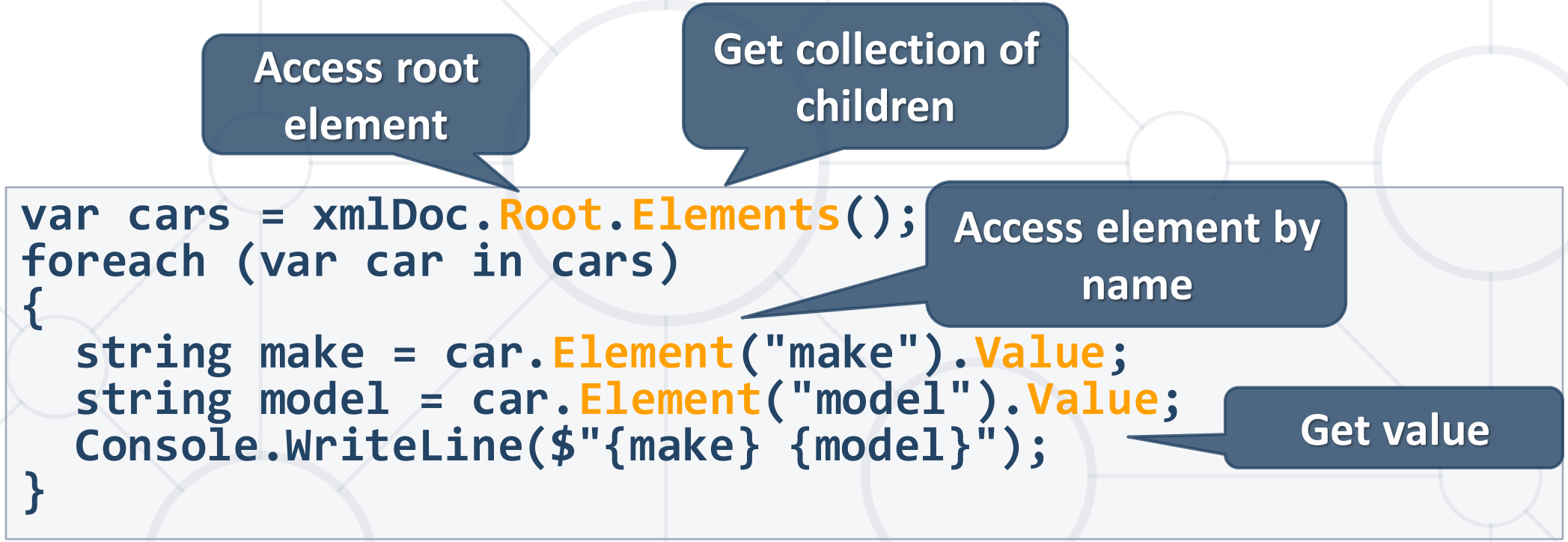
- To process an XML string:

```
string str =
@"<?xml version=""1.0""?>
<!-- comment at the root level -->
<Root>
    <Child>Content</Child>
</Root>";
XDocument doc = XDocument.Parse(str);
```

- Loading XML directly from file:

```
XDocument xmlDoc = XDocument.Load("../../books.xml");
```

Working with XDocument



Working with XDocument (2)

- Set an element value by name
  - If it doesn't exist, it will be **added**
  - If it is set to **null**, it will be **removed**

```
customer.SetElementValue("birth-date", "1990-10-04T00:00:00");
```

- Remove an element from its parent

```
var youngDriver = customer.Element("is-young-driver");
youngDriver.Remove();
```

Working with XDocument (3)

SoftUni Foundation

- Get or set an element attribute by name

```
customer.Attribute("name").Value
```

- Get a list of all attributes for an element

```
var attrs = customer.Attributes;
```

- Set an attribute value by name
  - If it doesn't exist, it will be **added**
  - If it is set to **null**, it will be **removed**

```
customer.SetAttributeValue("age", "21");
```

LINQ to XML – Searching with LINQ

- Searching in XML with LINQ is like searching with LINQ in array

```
XDocument xmlDoc = XDocument.Load("cars.xml");
var cars = xmlDoc.Root.Elements()
  .Where(e => e.Element("make").Value == "Opel" &&
              (long)e.Element("travelled-distance") >= 300000)
  .Select(c => new
    {
      Model = c.Element("model").Value,
      Traveled = c.Element("travelled-distance").Value
    })
  .ToList();
foreach (var car in cars)
  Console.WriteLine(car.Model + " " + car.Traveled);
```

Creating XML with XElement

- XDocuments can be composed from XElements and XAttributes

```
<books>
  <book>
    <author>Don Box</author>
    <title lang="en">Essential .NET</title>
  </book>
</book>
```

```
XDocument xmlDoc = new XDocument();
xmlDoc.Add(
  new XElement("books",
    new XElement("book",
      new XElement("author", "Don Box"),
      new XElement("title", "ASP.NET", new XAttribute("lang",
"en"))
))));
```

Add as root

Added with value

Optional attribute

Serializing XML to File

SoftUni Foundation

- To flush a XDocument to file with default settings:

```
xmlDoc.Save("myBooks.xml");
```

- To disable automatic indentation:

```
xmlDoc.Save("myBooks.xml", SaveOptions.DisableFormatting);
```

- To serialize **any object** to file:

```
var serializer = new XmlSerializer(typeof(ProductDTO));
using (var writer = new StreamWriter("myProduct.xml");)
{
  serializer.Serialize(writer, product);
}
```

Deserialize XML from String XML

SoftUni Foundation

- To **deserialize** an object from a XML string

```
var serializer = new XmlSerializer(typeof(OrderDto[]), new
XmlRootAttribute("Orders"));

var deserializedOrders =
  (OrderDto[])serializer.Deserialize(new StringReader(xmlString));
```

- Specifying **root attribute** name

```
var attr = new XmlRootAttribute("Orders");
var serializer = new XmlSerializer(typeof(OrderDto[]), attr);

var deserializedOrders =
  (OrderDto[])serializer.Deserialize(new
StringReader(xmlString));
```

XML Attributes

Using xml attributes

XML Attributes

SoftUni Foundation

- We can use several attributes to control serialization to XML
  - **[XmlType("Name")]** – Specifies the type's **name** in XML
  - **[XmlAttribute("name")]** – Serializes as **XML Attribute**
  - **[XmlElement]** – Serialize as **XML Element**
  - **[XmlIgnore]** – **Do not** serialize
  - **[XmlArray]** – Serialize as an **array** of XML elements
  - **[XmlRoot]** – Specifies the **root** element name
  - **[XmlText]** – Serialize **multiple xml elements** on **one line**

XML Attributes: Example

- We can use several XML attributes to control serialization

```
[XmlType("Book")]
public class BookDto
{
 [XmlAttribute("name")]
 public string Name { get; }

 [XmlElement("Author")]
 public string Author { get; }

 [XmlIgnore]
 public decimal Price { get; }
}
```

XML Type name

Not serialized

```
<Book name="It">
  <Author>Stephen King</Author>
</Book>
<Book name="Frankenstein">
  <Author>Mary Shelley</Author>
</Book>
<Book name="Queen Lucia">
  <Author>E.F. Benson</Author>
</Book>
<Book name="Paper Towns">
  <Author>John Green</Author>
</Book>
```

XML Serialization

Live Demo

Summary

- **XDocument** is a system object for working with XML in .NET, which supports LINQ
- XML can be read and saved **directly to file**
- **XML Attributes** are easy way to describe the **XML file**

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

INDEAVR
Serving the high achievers

INFRAGISTICS®

SUPERHOSTING.BG

SoftUni Organizational Partners

SoftUni Foundation

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license