XML Processing

Parsing XML

XDocument and LINQ-to-XML

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg/

Table of Contents

2

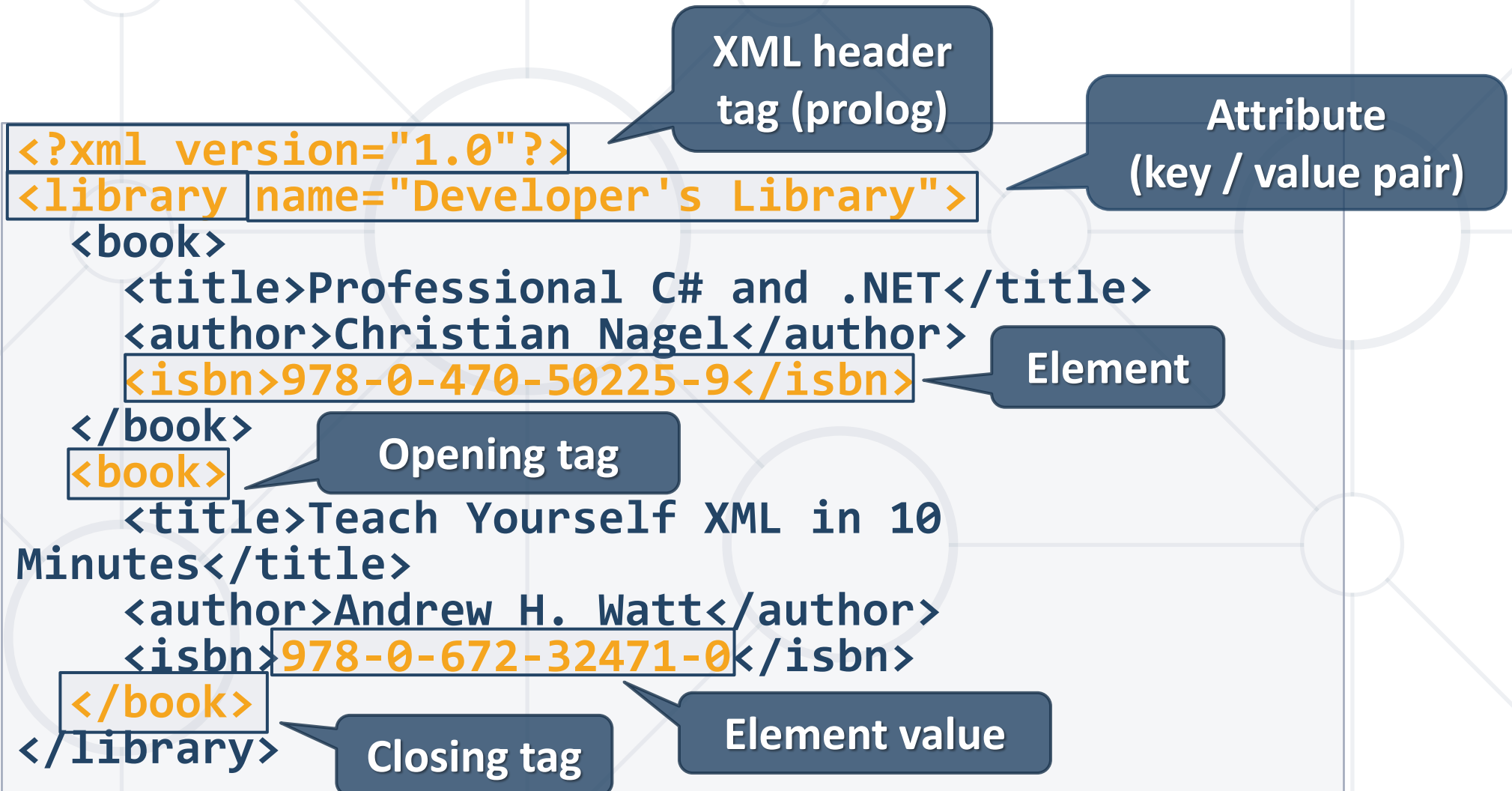# sli.do

#csharp-db

What is XML?

Format Description and Application

What is XML?

- E**X**tensible **M**arkup **L**anguage
  - **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

Software University

XML header tag (prolog)

Attribute (key / value pair)

Root (document) element

Element

Opening tag

Element value

Closing tag

```xml
<?xml version="1.0"?>
<library name="Developer's Library">
    <book>
        <title>Professional C# and .NET</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

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

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

- **Elements** – define the structure

- **Attributes** – element metadata

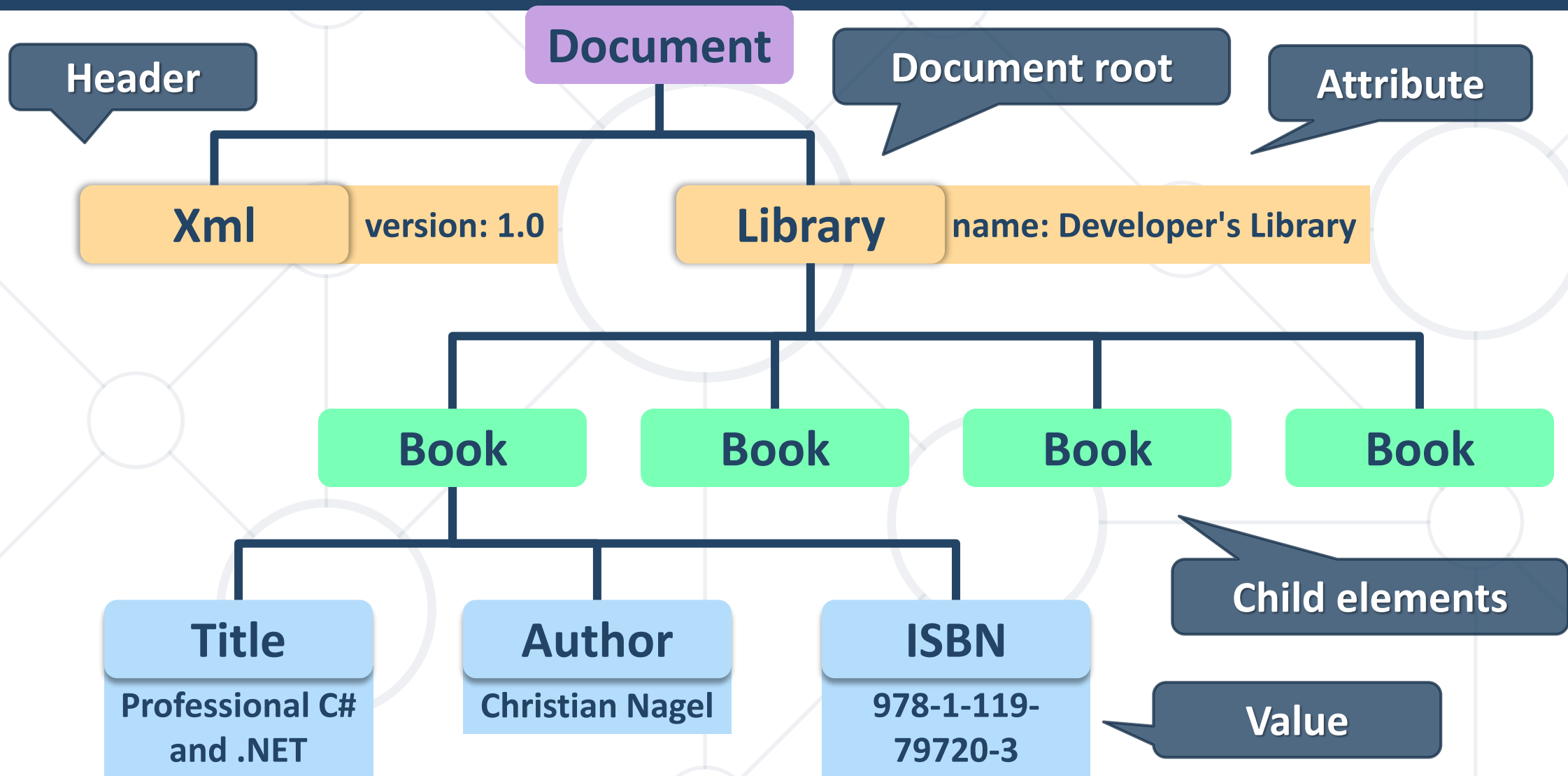- **Values** – actual data, that can also be nested elements

Element name   Attribute   Value

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

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

XML – Structure

XML and HTML

- **Similarities**
  - Both are **text based** notations
  - Both use **tags** and **attributes**
- **Differences**
  - 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

- 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

- 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.

XML vs JSON

- **XML**

  - XML data is typeless

  - All XML data should be string

  - Data needs to be parsed

  - Supports comments

  - Supports various encoding

- **JSON**

  - JSON object has a type

  - JSON types: string, number, array, Boolean

  - Data is accessible as JSON objects

  - Doesn't support comments

  - Supports only UTF-8 encoding

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

  - **XAttribute** – XML attributes 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");
```

15

Working with XDocument (1)

Access root element

Get collection of children

Access element by name

Get value

```csharp
var cars = xmlDoc.Root.Elements();
foreach (var car in cars)
{
    string make = car.Element("make").Value;
    string model = car.Element("model").Value;
    Console.WriteLine($"{make} {model}");
}
```

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)

- 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.Parse(e.Element("travelled-distance").Value) >= 30000)
    .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">ASP.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

- To flush an **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

- To **deserialize** an object from an 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

- 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**

- 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; }
}
```
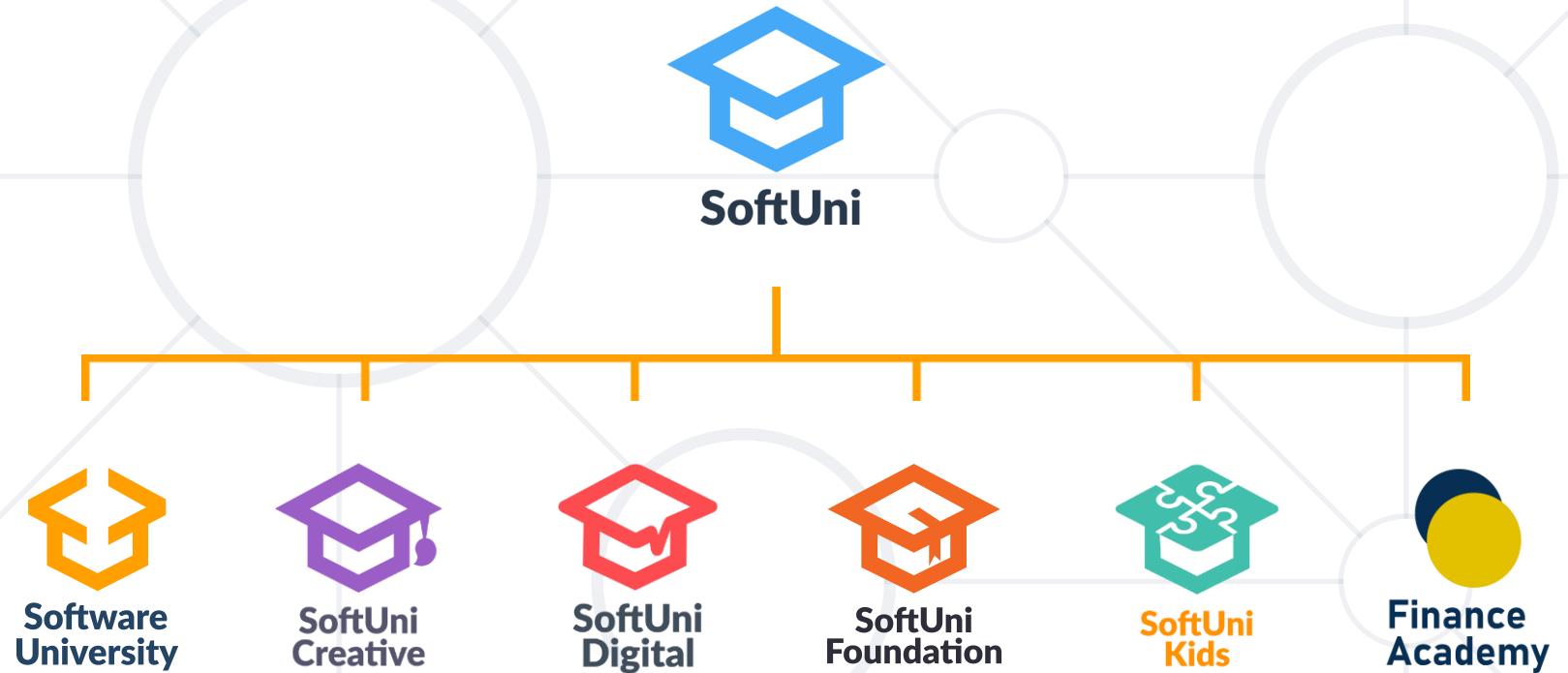
XML Type name

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>
```

25

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** can be serialized to and from **class**
- **XML Attributes** are easy way to describe the **XML file**

Questions?

SoftUni Diamond Partners

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers

  - softuni.bg, about.softuni.bg

- Software University Foundation

  - softuni.foundation

- Software University @ Facebook

  - facebook.com/SoftwareUniversity

- Software University Forums

  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg

- © Software University – https://softuni.bg