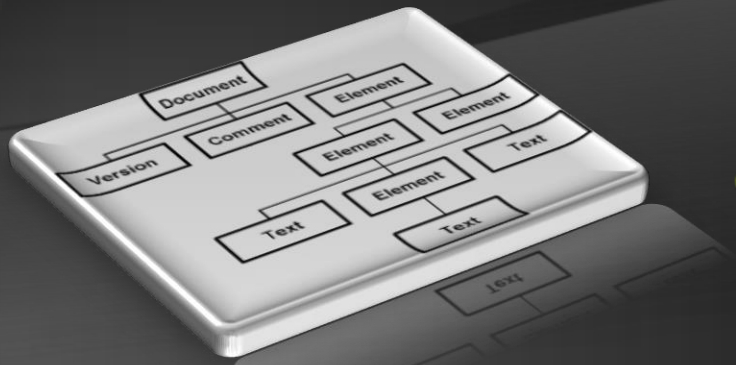
Telerik Academy



Processing XML in .NET

DOM Parser, Streaming Parser, XPath, LINQ to XML

Svetlin Nakov

Manager Technical Training

www.nakov.com

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

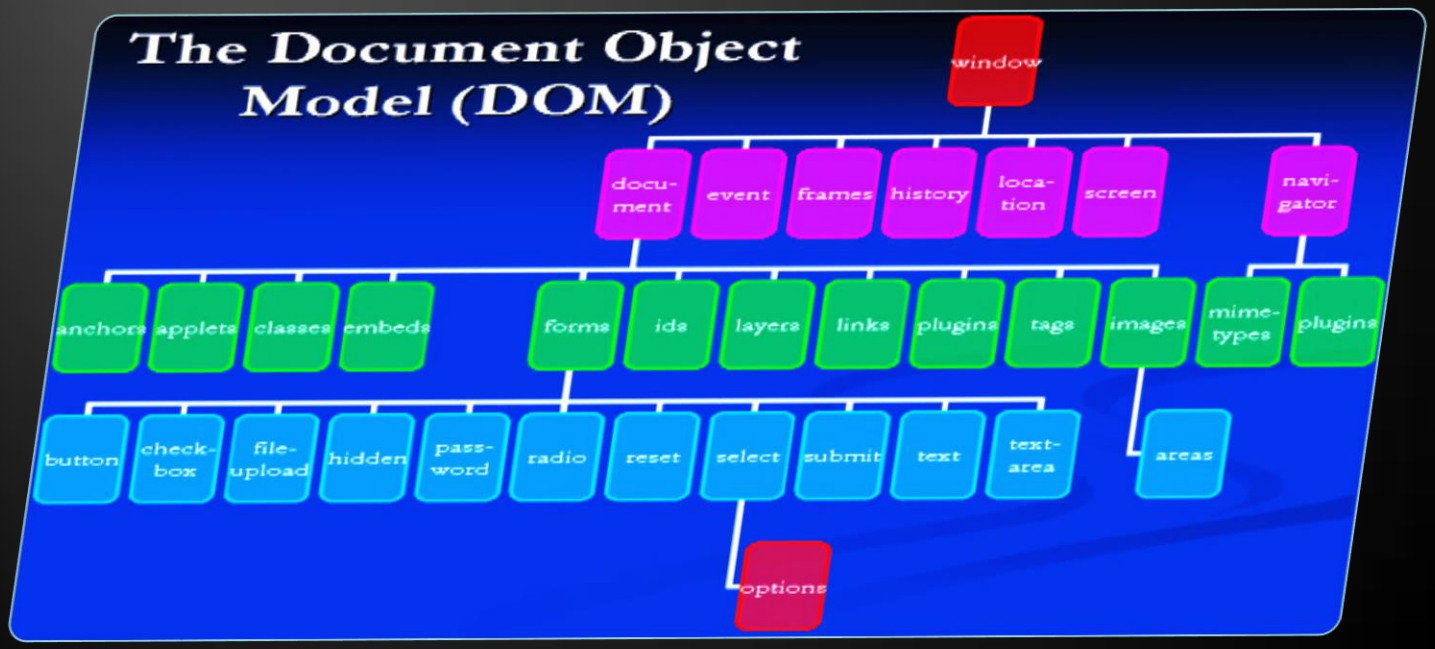
Table of Contents

1. Processing XML documents in .NET
   - Using the DOM Parser
   - Using the Streaming Parser
2. Using XPath to Search in XML Documents
3. Using LINQ-To-XML
4. XSL Transformation in .NET

Telerik Academy

The DOM Parser

Parsing XML Documents as DOM Trees



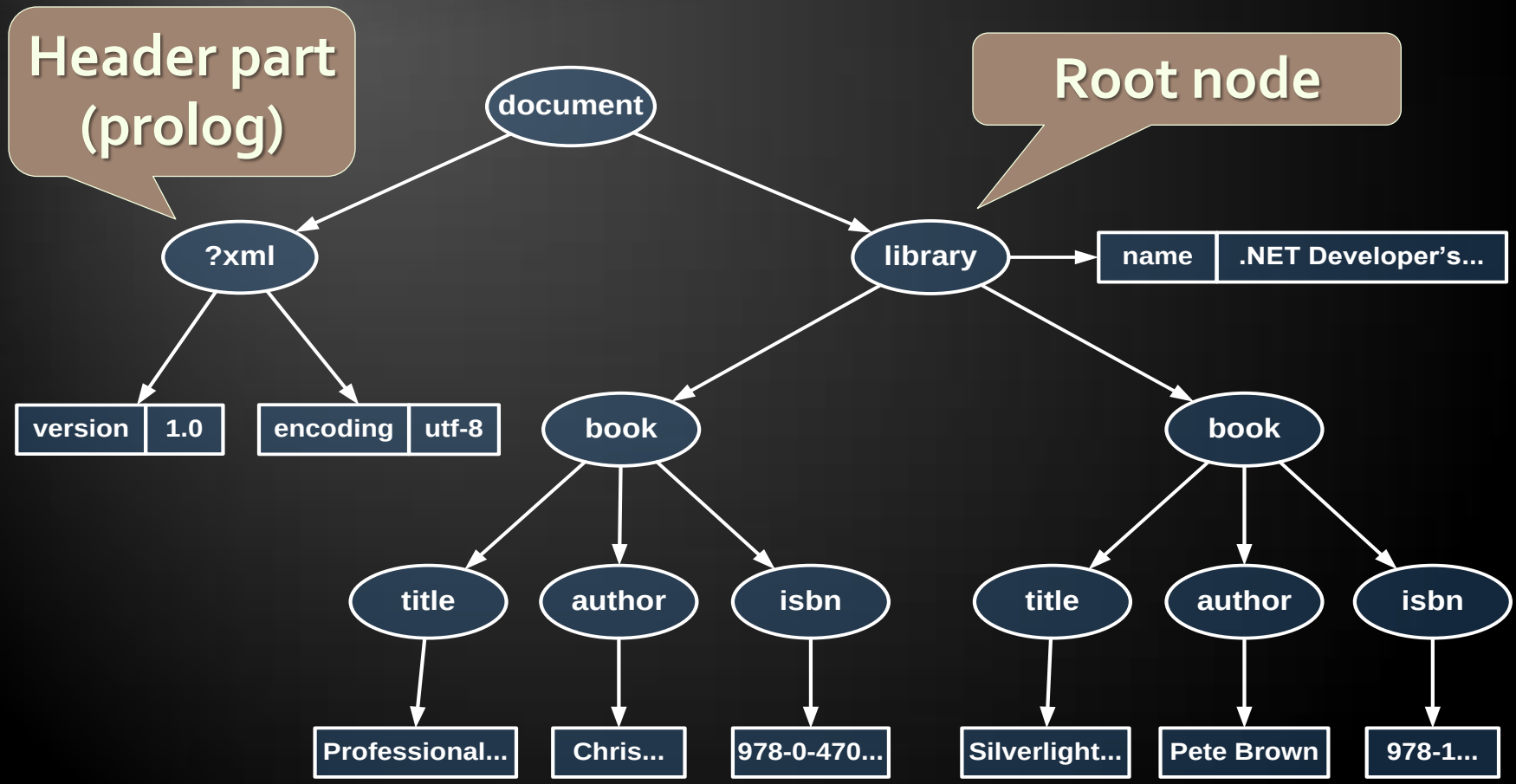
The DOM Parser

- The following XML document is given:

```
<?xml version="1.0"?>
<library name=".NET 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>Silverlight in Action</title>
    <author>Pete Brown</author>
    <isbn>978-1-935182-37-5</isbn>
  </book>
</library>
```

The DOM Parser (2)

- This document is represented in the in the memory as a DOM tree in the following way:



The DOM Parser – Example

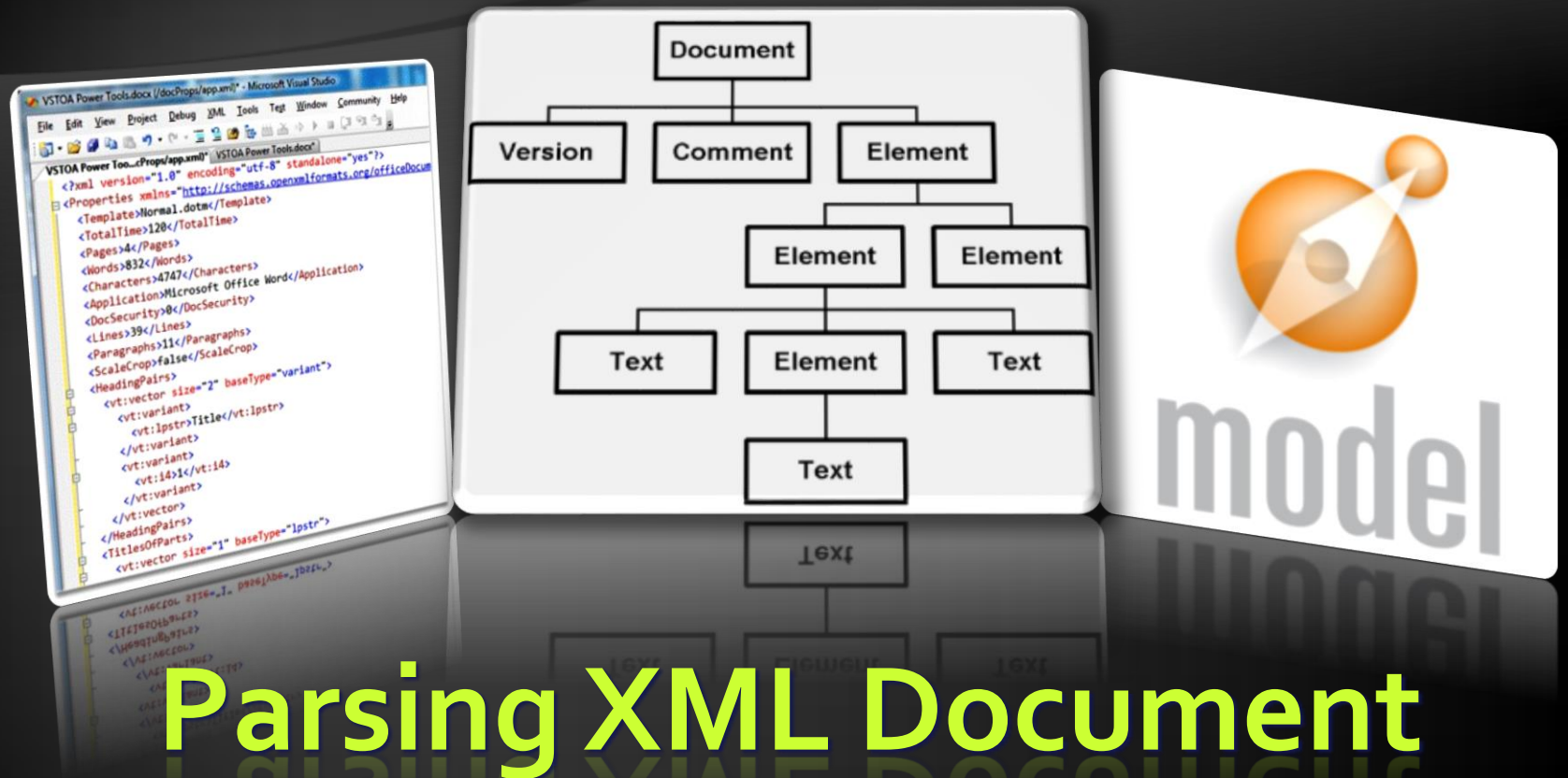
```
XmlDocument doc = new XmlDocument();
doc.Load("library.xml");

XmlNode rootNode = doc.DocumentElement;
Console.WriteLine("Root node: {0}", rootNode.Name);

foreach (XmlAttribute atr in rootNode.Attributes)
{
  Console.WriteLine("Attribute: {0}={1}",
    atr.Name, atr.Value);
}

foreach (XmlNode node in rootNode.ChildNodes)
{
  Console.WriteLine("\nBook title = {0}",
    node["title"].InnerText);
  Console.WriteLine("Book author = {0}",
    node["author"].InnerText);
  Console.WriteLine("Book isbn = {0}",
    node["isbn"].InnerText);
}
```

Telerik Academy



Parsing XML Document with the DOM Parser

Live Demo

Classes to Work with DOM

- The DOM parser provides few important classes:
  - `XmlDocument`
    - Represents the DOM tree
    - Usually contains two elements:
      - Header part (prolog)
      - The root element of the XML document
  - `XmlNode`
    - Abstract base class for all nodes in a DOM tree

Telerik Academy

Classes to Work with DOM (2)

- `XmlElement`
  - Represents a XML element
- `XmlAttribute`
  - Represents an attribute of an XML tag (couple name-value)
- `XmlAttributeCollection`
  - List of XML attributes attached to an element
- `XmlNodeList`
  - List of XML DOM nodes

The XmlNode Class

- The class `System.Xml.XmlNode`:
  - Fundamental for the DOM processing
  - Represents a base node
  - Its inheritor-classes are:
    - `XmlDocument`, `XmlElement`, `XmlAttribute`, `XmlDeclaration`, …
  - Allows navigation in the DOM tree:
    - `ParentNode` – returns the parent node (`null` for the root)

The XmlNode Class (2)

- `PreviousSibling` / `NextSibling` – returns the left / right node to the current
- `FirstChild` / `LastChild` – returns the first / last child of the current node
- `Item` (indexer `[]` in C#) – returns the child of the current node by its name

- Working with the current node:
  - `Name` – returns the name of the node (element, attribute …)
  - `Value` – gets the node value

The XmlNode Class (3)

- `Attributes` – returns the node attributes as `XmlAttributeCollection`
- `InnerXml`, `OuterXml` – returns the part of the XML containing the current node
  - Respectively with or without the node itself
- `InnerText` – concatenation of the values of the node and its inheritors
- `NodeType` – returns the node type

The XmlNode Class (4)

- Changing of the current node:
  - `AppendChild(…)` / `PrependChild(…)`
    - Inserts new child after / before all other children of the current node
  - `InsertBefore(…)` / `InsertAfter(…)`
    - Inserts new child before / after given inheritor
  - `RemoveChild(…)` / `ReplaceChild(…)`
    - Removes / replaces given child

The XmlNode Class (5)

- `RemoveAll()` – deletes all children of the current node (element, attribute …)
- `Value`, `InnerText`, `InnerXml` – changes the value / text / XML text of the node

The XmlDocument Class

- The `System.Xml.XmlDocument`:
  - Contains XML document represented as DOM tree
  - Allows loading and saving XML document (from files, streams or strings)
    - `Load(…)`, `LoadXml(…)`, `Save(…)`
- Important properties, methods and events:
  - `DocumentElement` – returns the root element

The XmlDocument Class (2)

- Important properties, methods and events:
  - `PreserveWhitespace` – indicates if the white space to be kept during save or load
  - `CreateElement(…)`, `CreateAttribute(…)`, `CreateTextNode(…)` – creates new XML element, attribute or value for the element
  - `NodeChanged`, `NodeInserted`, `NodeRemoved` – events for tracking the changes in the document

Telerik Academy

Modifying XML Document Using the DOM Parser

Live Demo

Telerik Academy

The Streaming Parser

Using XmlReader and XmlWriter

SAX and StAX Parsers

- SAX-style parsers are 'push' parsers
  - 'Push' data to the application
    - Event handlers for processing tags, attributes, data, whitespace, etc.
- StAX-like parsers are 'pull' parsers
  - 'Pull' the information from the parser as needed
  - The application moves the document's cursor forward (like reading a stream)
  - `XmlReader` is StAX-style (streaming) parser

SAX and StAX Parsers and XmlReader

- In .NET stream processing of XML document is done with the `XmlReader` class
- `XmlReader` is abstract class, which:
  - Provides read-only access to the XML data
  - Works like a stream, but reads XML documents
    - Works in forward-only mode
  - Read at each step data (tags, attributes, text) can be extracted and analyzed

The XmlReader Class

- The `XmlReader` class – most important methods and properties:
  - `Read()` – reads next node from the XML document or returns `false` if no such node
  - `NodeType` – returns the type of the read node
  - `Name` – returns the name of the read node (name of the last read element, attribute, …)
  - `HasValue` – returns `true` if the node has value
  - `Value` – returns the node value

The XmlReader Class (2)

- `ReadElementString()` – reads the value (the text) from the last read element
- `AttributeCount`, `GetAttribute(…)` – extract the attributes of the current element

- `XmlReader` is an abstract class
  - Instantiated with the static method `Create(…)`
  - Implements `IDisposable`

```
using (XmlReader reader = XmlReader.Create("items.xml"))
{
  while (reader.Read()) { … }
}
```

Telerik Academy

Working with XmlReader

Live Demo

Working with XmlWriter

- The class `XmlWriter` creates XML documents
  - Works as a stream, but writes tags and data into XML documents
- Most important methods:
  - `WriteStartDocument()` – adds the prolog part in the beginning of the document (`<?xml …`)
  - `WriteStartElement(…)` – adds opening tag
  - `WriteEndElement()` – closes the last tag
  - `WriteElementString(…)` – adds an element by defined name and text value

Telerik Academy

Working with XmlWriter (2)

- WriteAttributeString(…) – adds an attribute to the current element
- WriteEndDocument() – closes all tags and flushes the internal buffer (by calling Flush())

- XmlWriter is an abstract class
  - Instantiated by that static method Create(…)
  - Implements IDisposable

```
using (var writer = new XmlTextWriter ("items.xml"))
{
  // Write nodes here
}
```

Telerik Academy

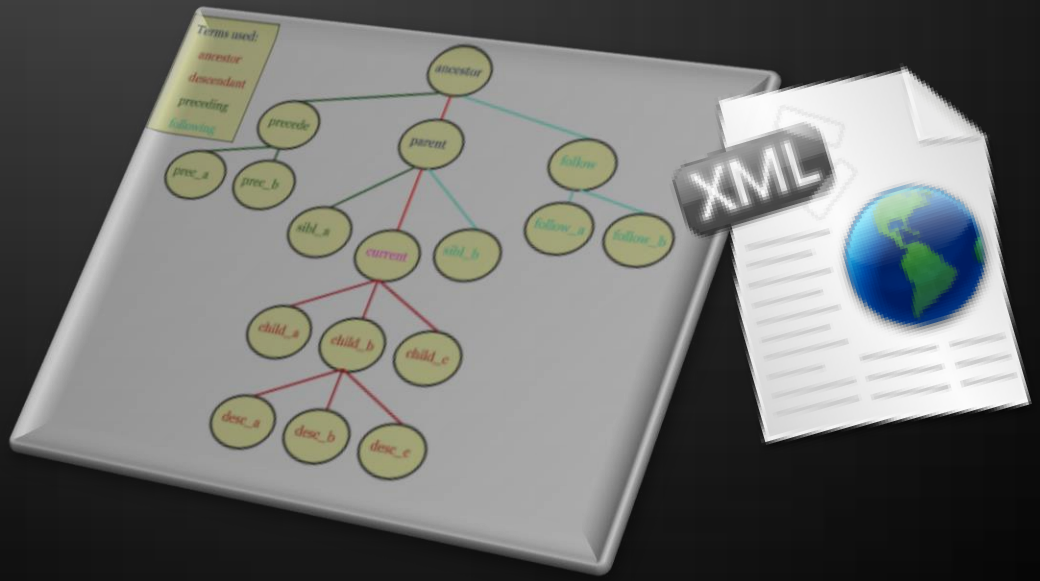
Working with XmlWriter

Live Demo

Telerik Academy

```
/catalog/cd[@price<10.80]
```

Using XPath



Using XPath in .NET

- XPath functionality is implemented in the `System.Xml.XPath` namespace
- XPath can be used directly from the class `XmlNode` (and all its inheritors) :
  - `SelectNodes(string xPathExpression)`
    - Returns list of all nodes matched by the specified XPath expression
  - `SelectSingleNode(string xPathExpression)`
    - Returns the first node matched by the specified XPath expression

Telerik Academy

XPath and XmlNode – Example

- Suppose we want to find the beer's nodes

```
<?xml version="1.0" encoding="windows-1251"?>
<items>
  <item type="beer">
    <name>Zagorka</name>
    <price>0.75</price>
  </item>
  <item type="food">
    <name>sausages</name>
    <price>0.48</price>
  </item>
  <item type="beer">
    <name>Kamenitza</name>
    <price>0.65</price>
  </item>
</items>
```

Telerik Academy

XPath and XmlNode – Example (2)

```
static void Main()
{
  XmlDocument xmlDoc = new XmlDocument();
  xmlDoc.Load("../../items.xml");
  string xPathQuery = "/items/item[@type='beer']";

  XmlNodeList beersList =
    xmlDoc.SelectNodes(xPathQuery);
  foreach (XmlNode beerNode in beersList)
  {
    string beerName =
      beerNode.SelectSingleNode("name");
    Console.WriteLine(beerName.InnerText);
  }
}
```

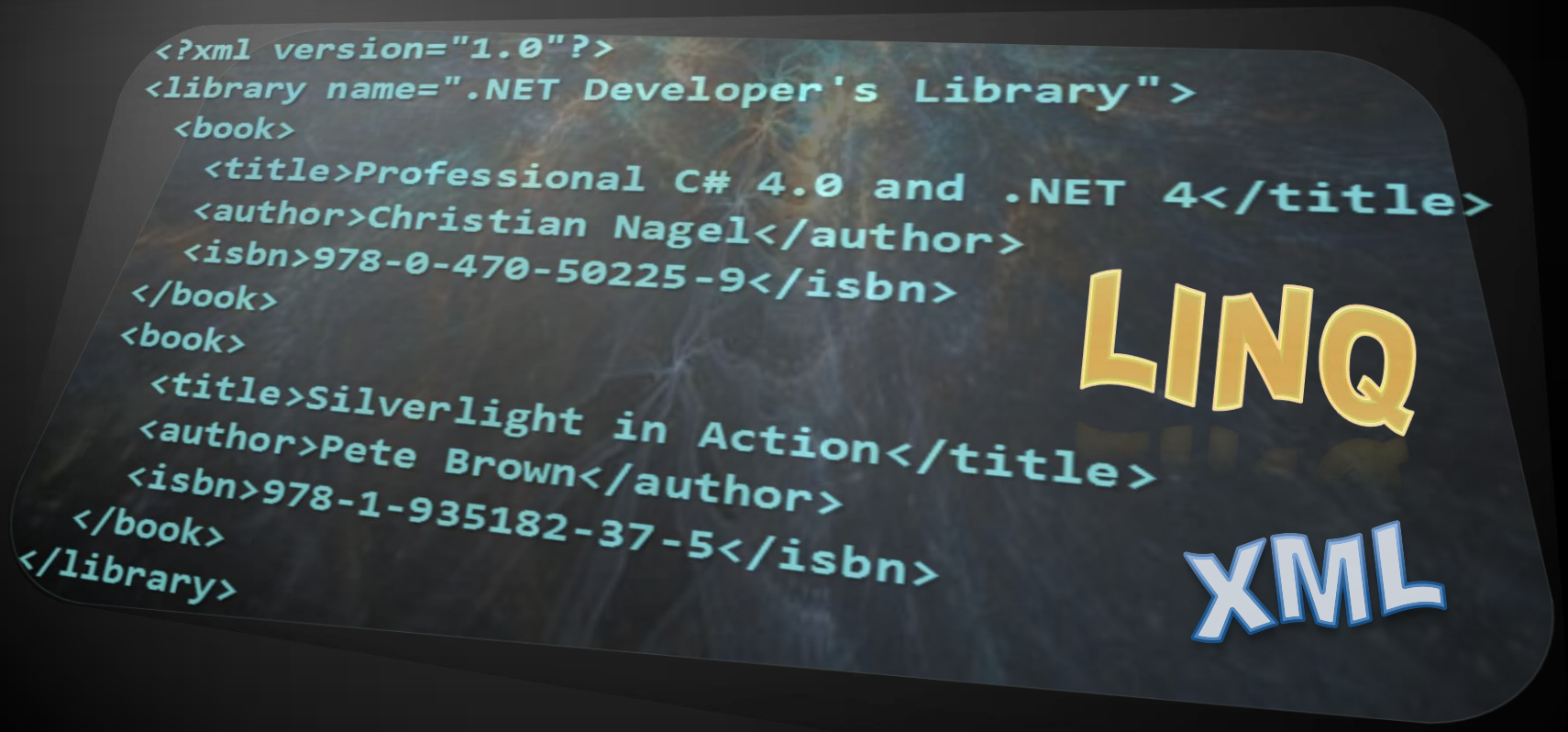
Telerik Academy

Searching with XPath in XML Documents

Live Demo

Telerik Academy

LINQ to XML



LINQ to XML

- LINQ to XML
  - Use the power of LINQ to process XML data
  - Easily read, write, modify, and search in XML documents
- Elements and attributes are no longer built-in the context of their parents
- LINQ to XML classes:
  - `XDocument` – represents a LINQ-enabled XML document (containing prolog, root element, …)

LINQ to XML Classes

- LINQ to XML classes:
  - `XElement` – represents an XML element
    - The most fundamental class in LINQ to XML
    - Important methods: `Parse`, `Save`, `RemoveAtributes`, `SetElementValue`, `SetAtributeValue`, `WriteTo`
  - `XAttribute` – name/ value attribute pair
  - `XName` – tag name + optional namespace – `{namespace}localname`
  - `XNamespace` – XML namespace

Telerik Academy

LINQ to XML – How Easy It Is!

```
<books>
    <book>
        <author>Don Box</author>
        <title>Essential .NET</title>
    </book>
</book>
```

- Need to create this XML fragment?

```
XElement books =
  new XElement("books",
    new XElement("book",
      new XElement("author", "Don Box"),
      new XElement("title", "ASP.NET")
    )
 );
```

Telerik Academy

Creating Document with LINQ To XML

Live Demo

Telerik Academy

LINQ to XML and Namespaces

- Elements in LINQ to XML have fully expanded names (namespace + name)
  - Easy to manipulate the elements in different namespaces

```
XNamespace ns = "http://linqinaction.net";
XNamespace anotherNS = "http://publishers.org";
XElement book = new XElement(ns + "book",
    new XElement(ns + "title", "LINQ in Action"),
    new XElement(ns + "author", "Manning"),
    new XElement(ns + "author", "Steve Eichert"),
    new XElement(ns + "author", "Jim Wooley"),
    new XElement(anotherNS + "publisher", "Manning")
);
```

Telerik Academy

LINQ to XML and Namespaces

Live Demo

LINQ to XML – Searching with LINQ

- Searching in XML with LINQ is as easy as searching within an array

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

var books =
  from book in xmlDoc.Descendants("book")
  where book.Element("title").Value.Contains("4.0")
  select new {
    Title = book.Element("title").Value,
    Author = book.Element("author").Value };

foreach (var book in books)
{
    Console.WriteLine(book);
}
```

Telerik Academy

Searching in XML with LINQ Queries

Live Demo

Telerik Academy

XSL Transformations

XSL Transformations

- XSL transformations (XSLT)
  - Convert a XML document to another XML document with different structure
  - Can convert XML to any text format
  - Can be used to transform XML documents to XHTML
- XSLT depends on XPath
  - For matching sections from the input document and replacing them with other text

Performing XSLT in .NET

- .NET has built-in XSLT engine
  - `System.Xml.Xsl.XslCompiledTransform`
- Methods:
  - `Load(…)`
    - Loads XSL shylesheet for transforming XML docs
  - `Transform(…)`
    - Performs transformation of given XML
    - Output is written to a XML file, stream or `XmlWriter`

Telerik Academy

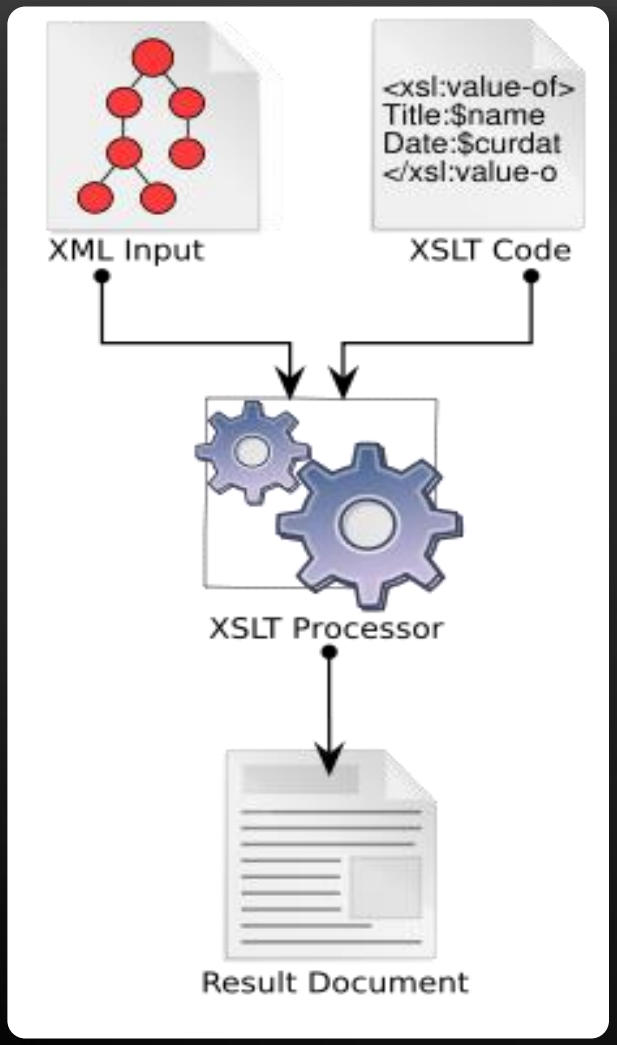
XSL Transformation – Example

- Transforming XML document by given XSL stylesheet:

```
using System;
using System.Xml.Xsl;

class XSLTransformDemo
{
  static void Main()
  {
    XslCompiledTransform xslt =
      new XslCompiledTransform();
    xslt.Load("catalog.xsl");
    xslt.Transform("catalog.xml", "catalog.html");
  }
}
```

Telerik Academy



Transforming XML by XSL Stylesheet

Telerik Academy

Processing XML in .NET

http://academy.telerik.com/student-courses/software-technologies/databases/

Exercises

1. Create a XML file representing catalogue. The catalogue should contain albums of different artists. For each album you should define: `name`, `artist`, `year`, `producer`, `price` and a list of `songs`. Each `song` should be described by `title` and `duration`.
2. Write program that extracts all different artists which are found in the `catalog.xml`. For each author you should print the number of albums in the catalogue. Use the DOM parser and a hash-table.
3. Implement the previous using XPath.
4. Using the DOM parser write a program to delete from `catalog.xml` all albums having price > 20.

Exercises (2)

5. Write a program, which using `XmlReader` extracts all song titles from `catalog.xml`.
6. Rewrite the same using `XDocument` and LINQ query.
7. In a text file we are given the name, address and phone number of given person (each at a single line). Write a program, which creates new XML document, which contains these data in structured XML format.
8. Write a program, which (using `XmlReader` and `XmlWriter`) reads the file `catalog.xml` and creates the file `album.xml`, in which stores in appropriate way the names of all albums and their authors.

Exercises (3)

9. Write a program to traverse given directory and write to a XML file its contents together with all subdirectories and files. Use tags `<file>` and `<dir>` with appropriate attributes. For the generation of the XML document use the class `XmlWriter`.
10. Rewrite the last exercises using `XDocument`, `XElement` and `XAttribute`.
11. Write a program, which extract from the file `catalog.xml` the prices for all albums, published 5 years ago or earlier. Use XPath query.
12. Rewrite the previous using LINQ query.

Exercises (4)

13. Create an XSL stylesheet, which transforms the file `catalog.xml` into HTML document, formatted for viewing in a standard Web-browser.
14. Write a C# program to apply the XSLT stylesheet transformation on the file `catalog.xml` using the class `XslTransform`.
15. * Read some tutorial about the XQuery language. Implement the XML to HTML transformation with XQuery (instead of XSLT). Download some open source XQuery library for .NET and execute the XQuery to transform the `catalog.xml` to HTML.

Exercises (5)

15. Using Visual Studio generate an XSD schema for the file `catalog.xml`. Write a C# program that takes an XML file and an XSD file (schema) and validates the XML file against the schema. Test it with valid XML catalogs and invalid XML catalogs.

Promotion

- Promotion – the beer became free (hurrah !!!)

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com
- Telerik Software Academy
  - academy.telerik.com
- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy
- Telerik Software Academy Forums
  - forums.academy.telerik.com