أكاديمية الحكومة الإلكترونية الفلسطينية

The Palestinian eGovernment Academy

www.egovacademy.ps

Tutorial III:

Process Integration and Service Oriented Architectures

# Session 3
# Xpath & Xquery


**Prepared By**

*Ismail Romi*
*Palestine Polytechnic University (PPU)*
*Email: ismailr@ppu.edu*


European Commission
TEMPUS


Reviewed by
Prof. Marco Ronchetti and Prof. Paolo Bouquet, Trento University, Italy

# About

This tutorial is part of the PalGov project, funded by the TEMPUS IV program of the Commission of the European Communities, grant agreement 511159-TEMPUS-1-2010-1-PS-TEMPUS-JPHES. The project website: www.egovacademy.ps

## Project Consortium:

- Birzeit University, Palestine (**Coordinator** )
- Palestine Polytechnic University, Palestine
- Palestine Technical University, Palestine
- Ministry of Telecom and IT, Palestine
- Ministry of Interior, Palestine
- Ministry of Local Government, Palestine
- University of Trento, Italy
- Vrije Universiteit Brussel, Belgium
- Université de Savoie, France
- University of Namur, Belgium
- TrueTrust, UK


**Coordinator:**
Dr. Mustafa Jarrar
Birzeit University, P.O.Box 14- Birzeit, Palestine
Telfax:+972 2 2982935 mjarrar@birzeit.edu

# © Copyright Notes

Everyone is encouraged to use this material, or part of it, but should properly cite the project (logo and website), and the author of that part.

No part of this tutorial may be reproduced or modified in any form or by any means, without prior written permission from the project, who have the full copyrights on the material.


Attribution-NonCommercial-ShareAlike
CC-BY-NC-SA

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.

# Tutorial Map

| Intended Learning Objectives |
|---|
| **A: Knowledge and Understanding** |
| 3a1: Demonstrate knowledge of the fundamentals of middleware. |
| 3a2: Describe the concept behind web service protocols. |
| 3a3: Explain the concept of service oriented architecture. |
| 3a4: Explain the concept of enterprise service bus. |
| 3a5: Understanding WSDL service interfaces in UDDI. |
| **B: Intellectual Skills** |
| 3b1: Design, develop, and deploy applications based on Service Oriented Architecture (SOA). |
| 3b2: use Business Process Execution Language (BPEL). |
| 3b3: using WSDL to describe web services. |
| **C: Professional and Practical Skills** |
| 3c1: setup, Invoke, and deploy web services using integrated development environment. |
| 3c2: construct and use REST and SOAP messages for web services communication. |
| **D: General and Transferable Skills** |
| d1: Working with team. |
| d2: Presenting and defending ideas. |
| d3: Use of creativity and innovation in problem solving. |
| d4: Develop communication skills and logical reasoning abilities. |

| Title | T | Name |
|---|---|---|
| Session0: Syllabus and overview | 0 | Aldasht |
| Sesson1: Introduction to SOA | 2 | Aldasht |
| Session2: XML namespaces & XML schema | 2 | Aldasht |
| Session 3: Xpath & Xquery | 4 | Romi |
| Session4: REST web services | 3 | M. Melhem |
| Session5: Lab2: Practice on REST | 3 | M. Melhem |
| Session 6: SOAP | 2 | Aldasht |
| Session 7: WSDL | **3** | Aldasht |
| Session8: Lab 3: WSDL practice | 3 | Aldasht |
| Session9: ESB | 4 | Aldasht |
| Session10: Lab4: Practice on ESB | 4 | Aldasht |
| Session11: integration patterns | 4 | M. Melhem |
| Session12: Lab5: integration patterns | 4 | M. Melhem |
| Session13: BPEL | 3 | Aldasht |
| Session14: Lab6: Practice on BPEL | 3 | Aldasht |
| Session15: UDDI | 2 | Aldasht |

# XPath: Content

- Ways of looking at an XML document.
- How to visualize XPath and how the component parts of XPath syntax fit together to enable you to navigate around the XPath data model.
- The XPath axes—the “directions” that are available to navigate around the XPath data model
- XPath 1.0 functions

# Ways of Looking at an XML Document

- The W3C has developed three specifications:
  - XPath
  - XML Document Object Model (DOM)
  - XML Information Set
- each of which represents a logical model of an XML document in similar but distinct ways.

# Modeling XML Documents

1. The XPath data model:
   - Represents most parts of a serialized XML document as a tree of nodes.
2. The Document Object Model:
   - Represents an XML document as a hierarchical tree of nodes.
3. The XMLInformation Set:
   - Represents an XMLdocument as a hierarchical tree but uses a different approach from both the XPath model and the DOM.
   - The XMLInformation Set recommendation is located at http://www.w3.org/TR/xml-infoset/.

# Visualizing XPath

- XPath can be considered as a street directions around the hierarchical tree of nodes that make up the *XPath data model* (Axes).
- XPath expressions starts from a standard point, (the root node).
- In XPath, the starting point is called the context.
- All legal XPath code can be called an expression.
- An XPath expression that returns a node-set is called a location path.

# Visualizing Xpath …Cont

- There are 13 directions that can be used.
- In XPath, a direction is called an axis.
- Example:
  - /Book/Chapter[@number=2]

# Understanding Context

- The context indicates the location of the node where a processor is currently situated.
- That node is called the context node.

# What Is a Node?

- A node is a representation in the XPath data model of a logical part of an XML document.
- Types:
  - ✓ Root node
  - ✓ Element node
  - ✓ Attribut node
  - ✓ Text node
  - ✓ Namespace node
  - ✓ Comment node
  - ✓ Processing Instruction node

# Root Node

- The root node represents the document itself, independent of any content.
- The root node is the apex of the hierarchy of nodes that represents an XML document.
- It has no name and cannot be seen when the document is serialized.
- The root node just serves as a starting point when navigating the document.
- The root element, is the first element in the document and is a child of the root node.

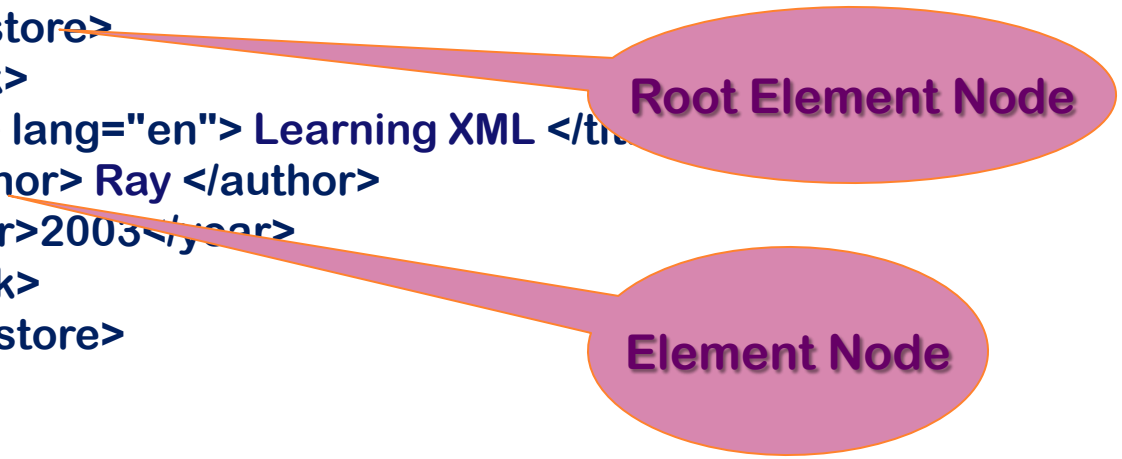
# Element Node

- Each element in an XML document is represented as an element node in the XPath data model.
- Element nodes have a name that consists of the namespace URI of the element and the local part of its name.

**The following is an XML document:**

```
<?xml version="1.0" encoding="ISO-8859-
1"?>

<bookstore>
  <book>
    <title lang="en"> Learning XML </ti
    <author> Ray </author>
    <year>2003</year>
  </book>
</bookstore>
```



# Attribute Node

- Each attribute in an XML document is represented in the XPath model as an attribute node.
- The element node with which the attribute node is associated is said to be the parent node of the attribute node.
- Attribute nodes have a name and a value.

***Example:***

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<bookstore>
 <book>
  <title lang="en"> Learning XML </title>
  <author> Ray </author>
  <year>2003</year>
 </book>
</bookstore>
```

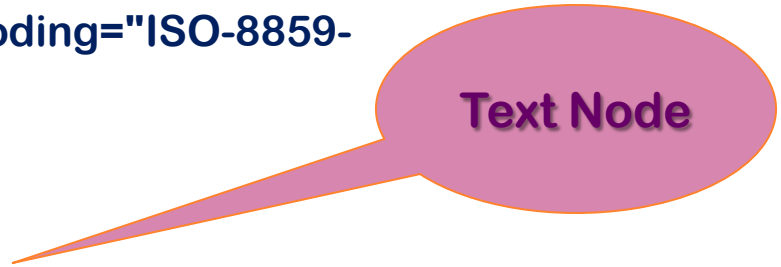
# Text Node

- Text content of an element node is represented in the XPath data model as a text node.
- The string value of a text node is its character data.
- A text node does not have a name.

***Example:***

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>
  <book>
    <title lang="en"> Learning XML </title>
    <author> Ray </author>
    <year>2003</year>
  </book>
</bookstore>
```



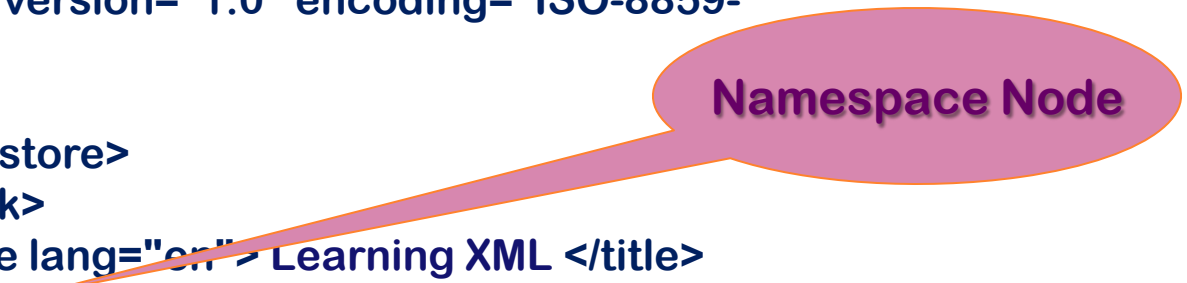
# Namespace Node

- Although a specific node can only belong to one namespace.
- Any number of in-scope namespaces can be in effect for the node.
- In-scope namespaces are those for which there exists a valid prefix to URI mapping or where a URI is associated with an empty prefix, the default namespace.

**The following is an XML document:**

```
<?xml version="1.0" encoding="ISO-8859-
1"?>

<bookstore>
 <book>
  <title lang="en"> Learning XML </title>
  <ppu:author> Ray </author>
  <year>2003</year>
 </book>
</bookstore>
```



# Comment Node

- A comment node represents a comment in the XML document.
- Comments in the document type declaration are not represented in the XPath data model.

***Example:***

```
<?xml version="1.0" encoding="ISO-8859-
1"?>

<bookstore>
<!-- This is An Example -->
 <book>
  <title lang="en"> Learning XML </title>
  <author> Ray </author>
  <year>2003</year>
 </book>
</bookstore>
```

# Processing Instruction Node (PI)

- A processing instruction node in the XPath model represents a processing instruction in the corresponding XML document.
- The name of a processing instruction node is its target.
- The string value of a processing instruction node is its content, excluding the target.
- ***Example Of PI's:***

•For MS Word 2003 the PI situated right after the XML declaration looks like this:

```
<?mso-application progid="Word.Document"?>
```

•For MS Excel 2003 the PI looks like this:

```
<?mso-application progid="Excel.Sheet"?>
```

# Location Paths

- An expression that specifies how to navigate an XPath tree from one node to another.
- A location path is composed of location steps, each of which is composed of:
  - An “axis,”
  - A “node test”
  - An optional “predicate.”
- To locate a specific node in an XML document, we put together multiple location steps, each of which refines the search.

# XPath 1.0 Axes

| Axis Name | Ordering | Description |
|---|---|---|
| **self** | none | The context node itself. |
| **parent** | reverse | The context node's parent, if one exists. |
| **child** | forward | The context node's children, if they exist. |
| **ancestor** | reverse | The context node's ancestors, if they exist. |
| **ancestor-or-self** | reverse | The context node's ancestors and also itself. |
| **descendant** | forward | The context node's descendants. |
| **descendant-or-self** | forward | The context node's descendants and also itself. |
| **following** | forward | The nodes in the XML document following the context node, not including descendants. |
| **following-sibling** | forward | The sibling nodes following the context node. |
| **preceding** | reverse | The nodes in the XML document preceding the context node, not including ancestors. |
| **preceding-sibling** | reverse | The sibling nodes preceding the context node. |
| **attribute** | forward | The attribute nodes of the context node. |
| **namespace** | forward | The namespace nodes of the context node. |

# Parent Axis

- Each element and attribute has one parent.
- Example

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
```

  - **book element** is the parent of the title, author, year, and price:

# Child Axis

- Element nodes may have zero, one or more children.
- Example

```
<book>
   <title>Harry Potter</title>
   <author>J K. Rowling</author>
   <year>2005</year>
   <price>29.99</price>
</book>
```

  - **title, author, year, and price elements** are all children of the book element:

# Siblings

- Nodes that have the same parent.
- Example:

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

  - **title, author, year, and price** elements are all siblings.

# Ancestors

- Node's parent, parent's parent, etc.
- Example:

```
<bookstore>
  <book>
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

  - **The ancestors** of the title element are the book element and the bookstore element.

# Descendants

- A node's children, children's children, etc.
- Example:

```
<bookstore>
  <book>
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

  - Descendants of the bookstore element are the book, title, author, year, and price elements.

# Location Path Expression

- An XPath expression returns either a node-set, a string, a Boolean, or a number.
- A location path can be absolute or relative.
  - An absolute location path starts with a slash ( / )
  - A relative location path does not.
- In both cases the location path consists of one or more **steps.**
- *Examples:*
- An absolute location path:
  /step/step/...
- A relative location path:
  step/step/...

# Location Path Expression…Cont

- The syntax for a location step is:

  axisname::nodetest[predicate]

# Location Path Expression…Cont

- Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    ……
   ……
<bookstore>
     <book>
       <title lang="en">Harry Potter</title>
       <author>J K. Rowling</author>
       <year>2005</year>
       <price>29.99</price>
     </book>
   </bookstore>
```

# Location Path Expression…Cont

**axisname::nodetest[predicate]**

| Example | Result |
|---|---|
| child::book | Selects all book nodes that are children of the current node |
| attribute::lang | Selects the lang attribute of the current node |
| child::* | Selects all children of the current node |
| attribute::* | Selects all attributes of the current node |
| child::text() | Selects all text child nodes of the current node |
| child::node() | Selects all child nodes of the current node |
| descendant::book | Selects all book descendants of the current node |
| ancestor::book | Selects all book ancestors of the current node |
| ancestor-or-self::book | Selects all book ancestors of the current node - and the current as well if it is a book node |
| child::*/child::price | Selects all price grandchildren of the current node |

# Location Path Expression…Cont

| Node Test | Description |
|---|---|
| * | Selects all nodes of the same principal node type. |
| **node()** | Selects all nodes, regardless of their type. |
| **text()** | Selects all text nodes. |
| **comment()** | Selects all comment nodes. |
| **processing-instruction()** | Selects all processing-instruction nodes. |
| *node name* | Selects all nodes with the specified *node name*. |

## Steps Operators:

- ✓slash (/)Separates location steps.
- ✓double-slash (//)Abbreviation for the location path

The Palestinian e-government academy

# Xpath Operators

| Operator | Description | Example | Return value |
|---|---|---|---|
| \| | Performs the union of two node-sets | //book \| //cd | Returns a node-set with all book and cd elements |
| + | Addition | 6+4 | 10 |
| - | Subtraction | 6-4 | 2 |
| * | Multiplication | 6*4 | 24 |
| div | Division | 8 div 4 | 2 |
| = | Equal | price=9.80 | true if price is 9.80 |
| != | Not equal | price!=9.80 | false if price is 9.80 |
| > | Less than | price<9.80 | true if price is 9.00 |
| <= | Less than or equal to | price<=9.80 | true if price is 9.00 |
| < | Greater than | price>9.80 | true if price is 9.90 |
| >= | Greater than or equal to | price>=9.80 | true if price is 9.90 |
| or | or | price=9.80 or price=9.70 | true if price is 9.80 |
| and | and | price>9.00 and price<9.90 | true if price is 9.80 |
| mod | Modulus (division remainder) | 5 mod 2 | 1 |

# node-set functions…Examples

| Node-set Functions | Description |
|---|---|
| **last()** | Returns the last node in the node-set. |
| **position()** | Returns the position number of the current node in the node-set being tested. |
| **count(** *node-set* **)** | Returns the number of nodes in *node-set*. |
| **id(** *string* **)** | Returns the element node whose ID attribute matches the value specified by argument *string*. |
| **local-name(** *node-set* **)** | Returns the local part of the expanded-name for the first node in *node-set*. |
| **namespace-uri(** *node-set* **)** | Returns the namespace URI of the expanded-name for the first node in *node-set.* |
| **name(** *node-set* **)** | Returns the qualified name for the first node in *node-set*. |

# Examples

- head/title[ last() ]
  - select the last title element node contained in the head element node
- book[ position() = 3 ]
  - select the third book element of the context node. The location path
  - ……
  - …..

# Databases: Xquery  (the XML Query Language)

- Why XQuery was created.
- How to get started with Xquery.
- How to query an XMLdocument using Xquery.
- XQuery data model.
- XQuery functions.

# XQuery

- XQuery is to XML what SQL is to database tables.
- XQuery is designed to query XML data - not just XML files, but anything that can appear as XML, including databases.
- Xquery is designed around a data model that has the property to be serialized as XML.
- XML data requires many of the features already available in DBMS (indexing, security….).
- XQuery is compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema.
- XQuery 1.0 became a W3C Recommendation January 23, 2007.

# XQuery Tools

- XQuery is still relatively new.
- A large number of software companies and independent developers have developed partial or more complete implementations of XQuery.
- W3C updates a web page where links to XQuery implementations and other sources of useful XQuery information are included (www.w3.org/XML/Query).

# XQuery Tools…Cont

Example:

- Saxon Xquery engine.
- X-Hive Database
- Tamino Database
- Microsoft SQL Server 2005
- Oracle
- Stylus Stodio

# XQuery - Examples of Use

- XQuery can be used to:
  - Extract information to use in a Web Service
  - Generate summary reports
  - Transform XML data to XHTML
  - Search Web documents for relevant information

# Extracting Data From XML Document

- Open the xml document:
  - The doc() function is used to open the xml document.
- Path Expressions:
  - XQuery uses path expressions to navigate through elements in an XML document.
- Predicates:
  - XQuery uses predicates to limit the extracted data from XML documents.
- Functions:
  - XQuery uses functions to extract data from XML documents.

# doc() Function

- Used to open the XML document.
- Exampl books.xml
- Retrieving the previous document:
  - doc(books.xml)
- Retrieving part of the document:
  - You have to use the Xpath expressions.
  - doc("books.xml")/bookstore/book/title

# Predicates

- XQuery uses predicates to limit the extracted data from XML documents.
- Example:
  - doc("books.xml")/bookstore/book[price<30]

# XQuery Basic Syntax Rules

- XQuery is case-sensitive
- XQuery elements, attributes, and variables must be valid XML names
- An XQuery string value can be in single or double quotes
- An XQuery variable is defined with a $ followed by a name, e.g. $bookstore
- XQuery comments are delimited by (: and :)
- *Example:*

  (: XQuery Comment :)

# XQuery FLWOR Expressions (for, let, where, order by, return)

- **for** clause selects all elements.
- **where** clause selects only elements with a specified predicate.
- **order by** clause defines the sort-order.
- **return** clause specifies what should be returned.
- **let** clause allows variable assignments.

# for , return expressions

- Used for looping
- *Example:*

```
for $x in doc("books.xml")/bookstore/book
return $x/title
```

- This for:
  - loops all bookstore/book
  - returns the title element
- $x: variable name

# Filtering with where clause

- where used in for expression to filter the returned values.
- Example:

```
for $x in doc("books.xml")/bookstore/book
 where $x/price>30
 return $x/title
```

# Sorting using order by clause

- Used to sort the output in a specified order.
- Example:

```
for $x in doc("books.xml")/bookstore/book
 where $x/price>30
 order by $x/title
 return $x/title
```

# The let Clause

- The let clause:
  - allows variable assignments
  - avoids repeating the same expression many times.
  - does not result in iteration.
- Example:

```
let $x := "Ahmad"
return <name>{$x}</name>
```

Output:

```
<name> Ahmad </name>
```

# If-Then-Else

- An XQuery Conditional Expressions.
- Example:

```
for $x in doc("books.xml")/bookstore/book
 return if ($x/@category="CHILDREN")
            then <child>{data($x/title)}</child>
            else <adult>{data($x/title)}</adult>
```

- Notes:
  - $x: refers to variable name x
  - @ refers to an attribute
  - parentheses around the if expression are required.
  - else is required, but it can be just else ().

# XQuery Adding Elements and Attributes

- You can add elements and attributes to the output document.

Example:

```
for $x in doc("books.xml")/bookstore/book/title
return $x, <name> ali </name>
```

- This example adds name element to the output stream.

# Xquery Functions

- XQuery includes over 100 built-in functions.
- There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more.
- Functions found in the w3c web site: www.w3.org/tr/xpath-functions

# Examples of built-in Function

```
<name>{uppercase($booktitle)}</name>

for $x in doc("books.xml")/bookstore/book
order by $x/title
return $x/data(title)
```

Note: you have to practice w3c Xquery functions.

# Xquery and HTML

- You can generate an HTML document using Xquery.
- You can add elements
- You can add attribute.

# Example1:

## Xquery expression:

```
<ul>
    {
    for $x in doc("books.xml")/bookstore/book/title
    order by $x
    return <li>{$x}</li>
    }
    </ul>
```

## Output:

```
<ul>
    <li><title lang="en">Everyday Italian</title></li>
    <li><title lang="en">Harry Potter</title></li>
    <li><title lang="en">Learning XML</title></li>
    <li><title lang="en">XQuery Kick Start</title></li>
    </ul>
```

# Example2

- Xquery expression:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

- Out put / HTML

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

# Example 3

- **Xquery:**

```
<html>
    <body>

    <h1>Bookstore</h1>

    <ul>
    {
    for $x in doc("books.xml")/bookstore/book
    order by $x/title
    return <li>{data($x/title)}. Category: {data($x/@category)}</li>
    }
    </ul>

    </body>
    </html>
```

# Example 3….Cont

- **Out put:**

```
<html>
<body>

<h1>Bookstore</h1>

<ul>
<li>Everyday Italian. Category: COOKING</li>
<li>Harry Potter. Category: CHILDREN</li>
<li>Learning XML. Category: WEB</li>
<li>XQuery Kick Start. Category: WEB</li>
</ul>

</body>
</html>
```

# References


1. Hunter, H, Rafter, J., Fawcett, J., Vlist, E., Ayers, D., Duckett, J., Watt, A., McKinnon,L., (2007), "**Beginning XML**", 4th Ed.,*Wiley Publishing Inc: Indiana, USA.*
2. Ray, E., (2003), "**Learning XML**", 2nd Ed., *O'Rreilly Media Inc.: USA*.
3. http://www.w3.org
4. http://www.w3schools.com
5. http://www.xml.com
6. http://www.xml.org

# Thanks

*Ismail Romi*