XML and XPath with PHP

TobiasSchlitt <toby@php.net>

IPC SE 2009

2009-05-29

About me

- Tobias Schlitt <toby@php.net>
- PHP since 2001
- Freelancing consultant
- Qualified IT Specialist
- Studying CS at TU Dortmund

  (expect to finish this year)
- OSS addicted
  - PHP
  - eZ Components
  - PHPUnit

And who are you?

- What is your name?
- Where are you from?
- What is your business?
- What are your experiences with XML / XPath?
- What do you expect from this workshop?

Overview

Outline

Outline - XML

What is XML?

- General specification for creating markup languages
- Data exchange between computer systems
  - System independent
  - Human readable
  - Most used on the web
  - Successor of SGML
- W3C recommendation
  - 1.0 1998 (last update 2008-11-26)
  - 1.1 2004 (last update 2006-08-16)

XML languages by example

XHTML
- XML variant of the popular HTML

RSS
- Really Simply Syndication
- Provide news / updates / ... of websites
- Read by special clients
- Aggregation on portals / planets

SVG
- Scalable Vector Graphics
- Describe vector graphics in XML
- Potentially interactive / animated (via ECMAScript)

Related technologies - Schemas

Schema

A schema defines the structure XML instance documents.

XMLSchema

- Written in XML
- W3C recommendation
- Popular

RelaxNG

- 2 syntax variants
  - XML based
  - Short plain text based
- OASIS / ISO standard
- Popular

DTD

- Plain text
- W3C recommendation
- Deprecated

Related technologies - Querying

Query

A query extracts a sub-set of information from a data source.

XPath

- W3C recommendation
- Navigation in XML documents
- more on that later...

XQuery

- Functional programming language
- Allows complex queries

Outline - XML

An XML document

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

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Preamble

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

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Element nodes

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

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Attribute nodes

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

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Atomic value nodes

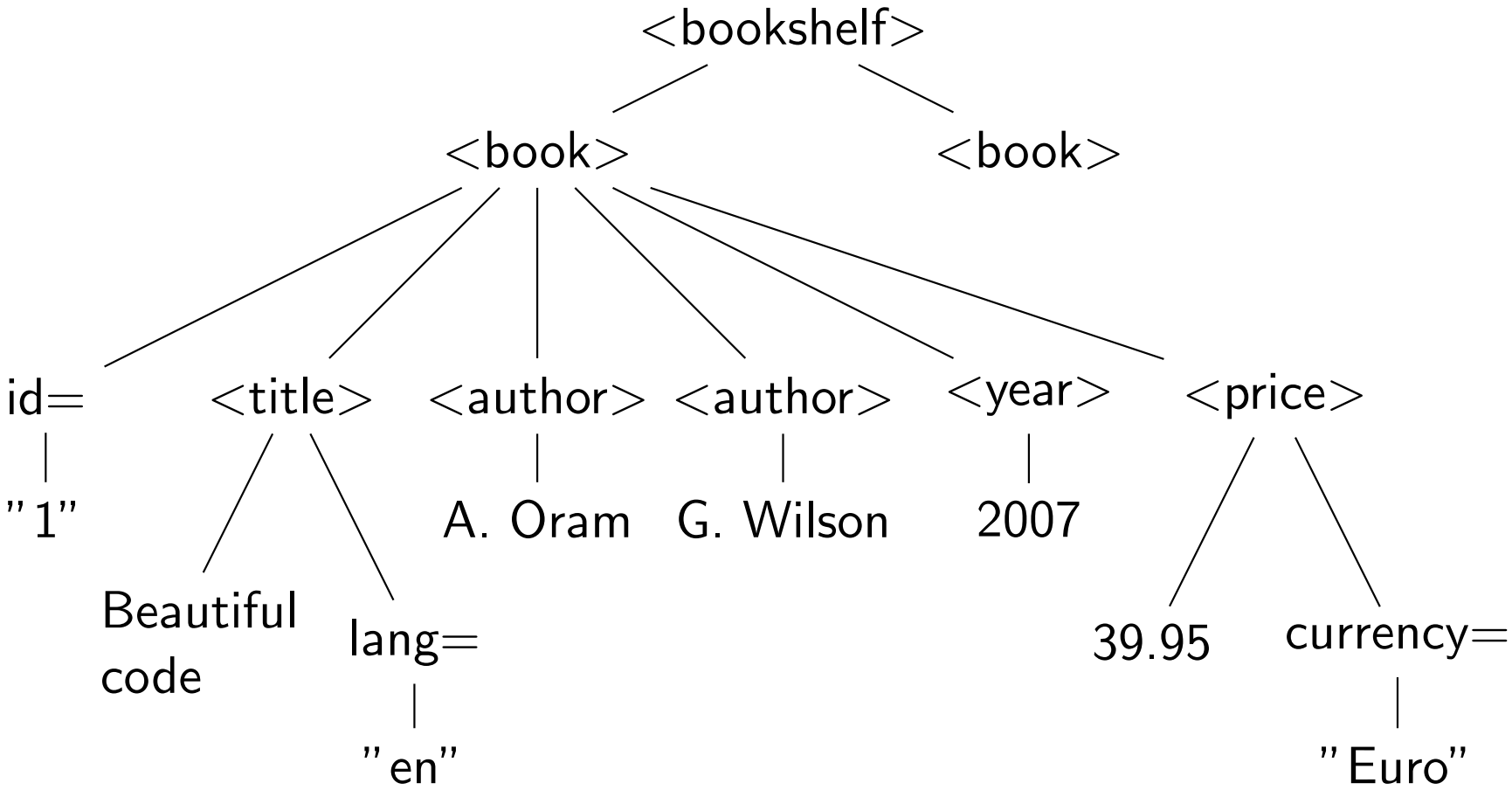
```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Document element

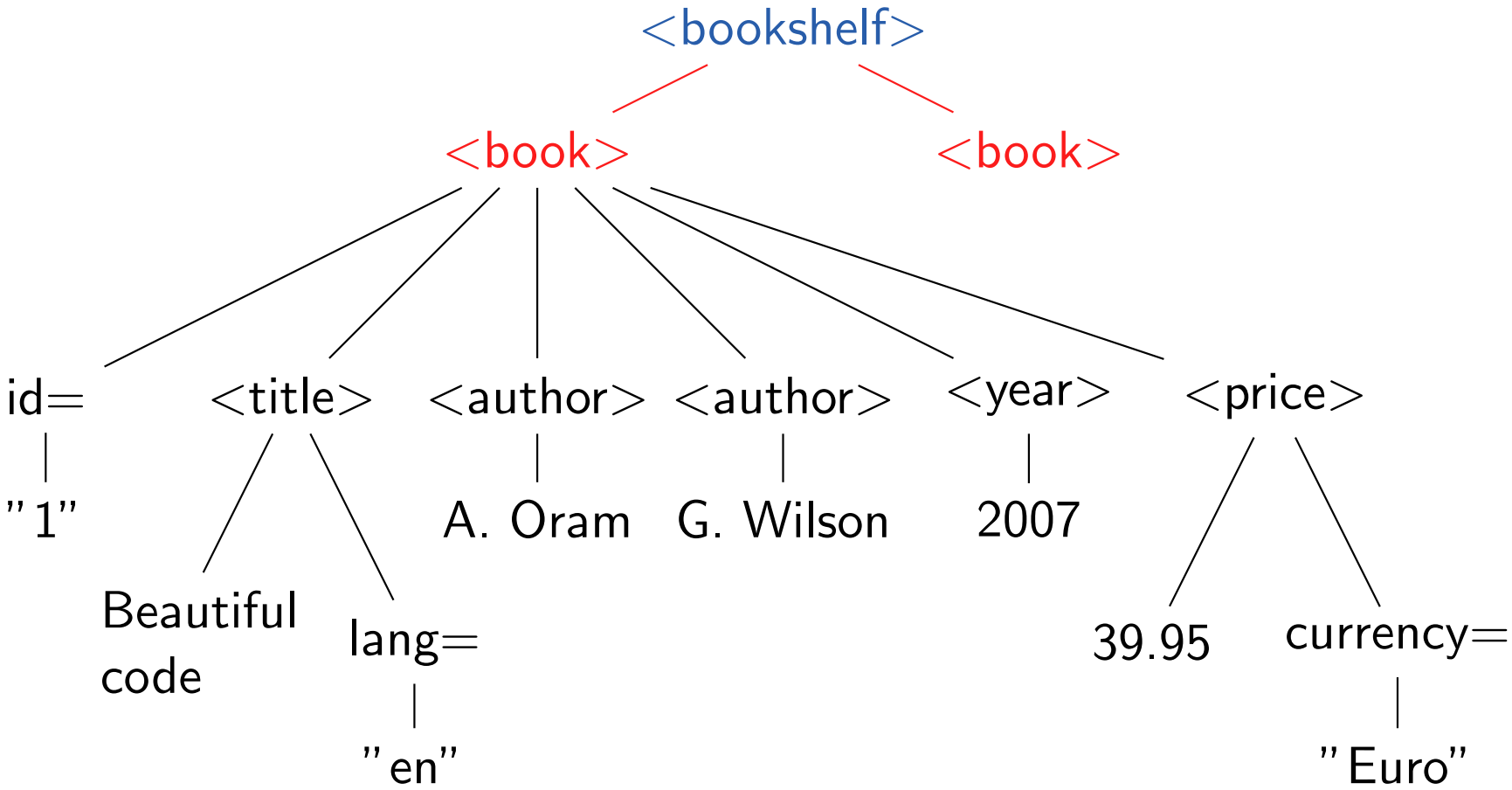
```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>

</bookshelf>
```

Outline - XML

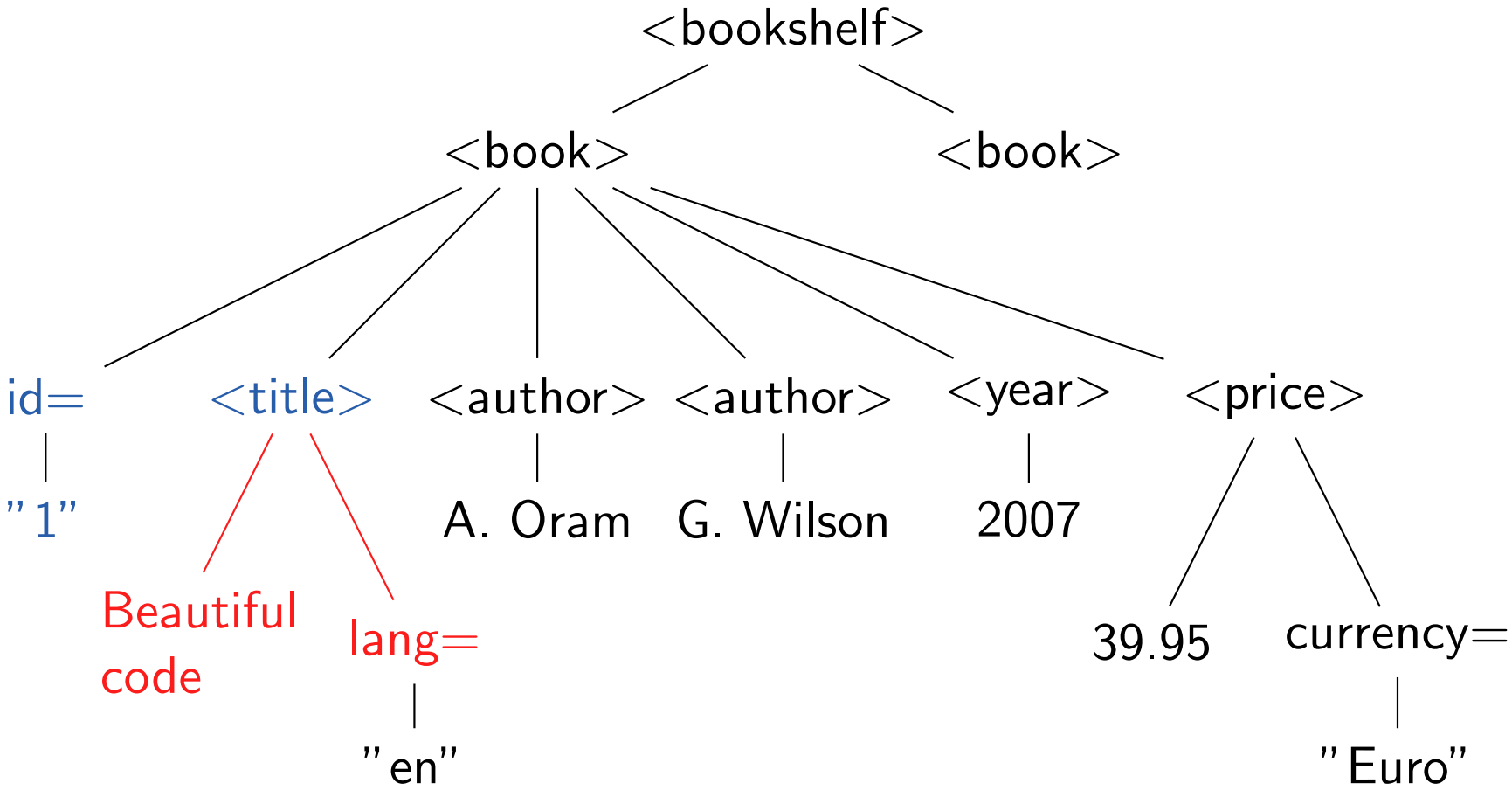
The XML tree


<bookshelf>
<book>
<book>
id=
"1"
<title>
Beautiful
code
lang=
"en"
<author>
A. Oram
<author>
G. Wilson
<year>
2007
<price>
39.95
currency=
"Euro"

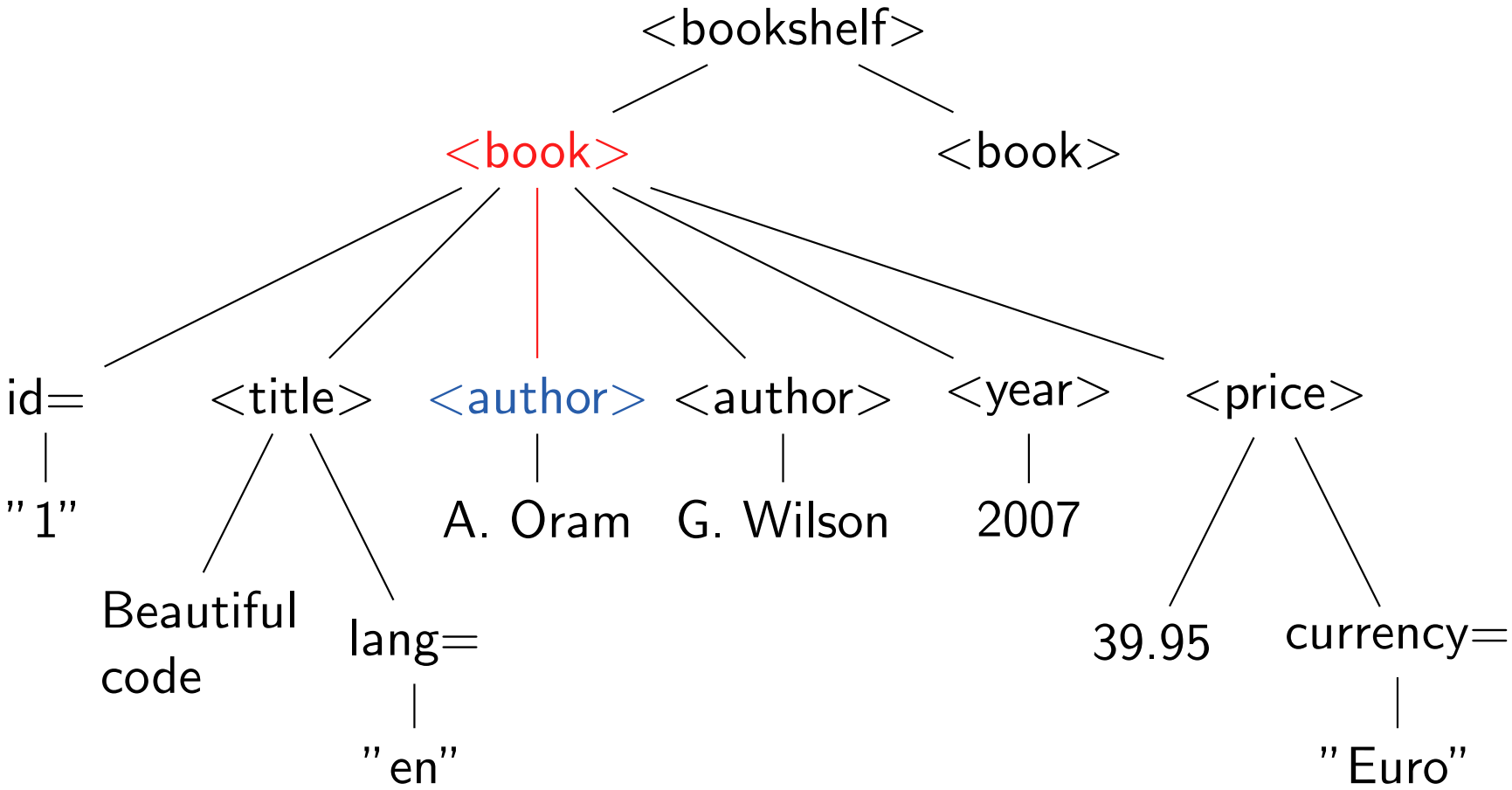
Children I


<bookshelf>
<book>
<book>
id=
"1"
<title>
Beautiful
code
lang=
"en"
<author>
A. Oram
<author>
G. Wilson
<year>
2007
<price>
39.95
currency=
"Euro"

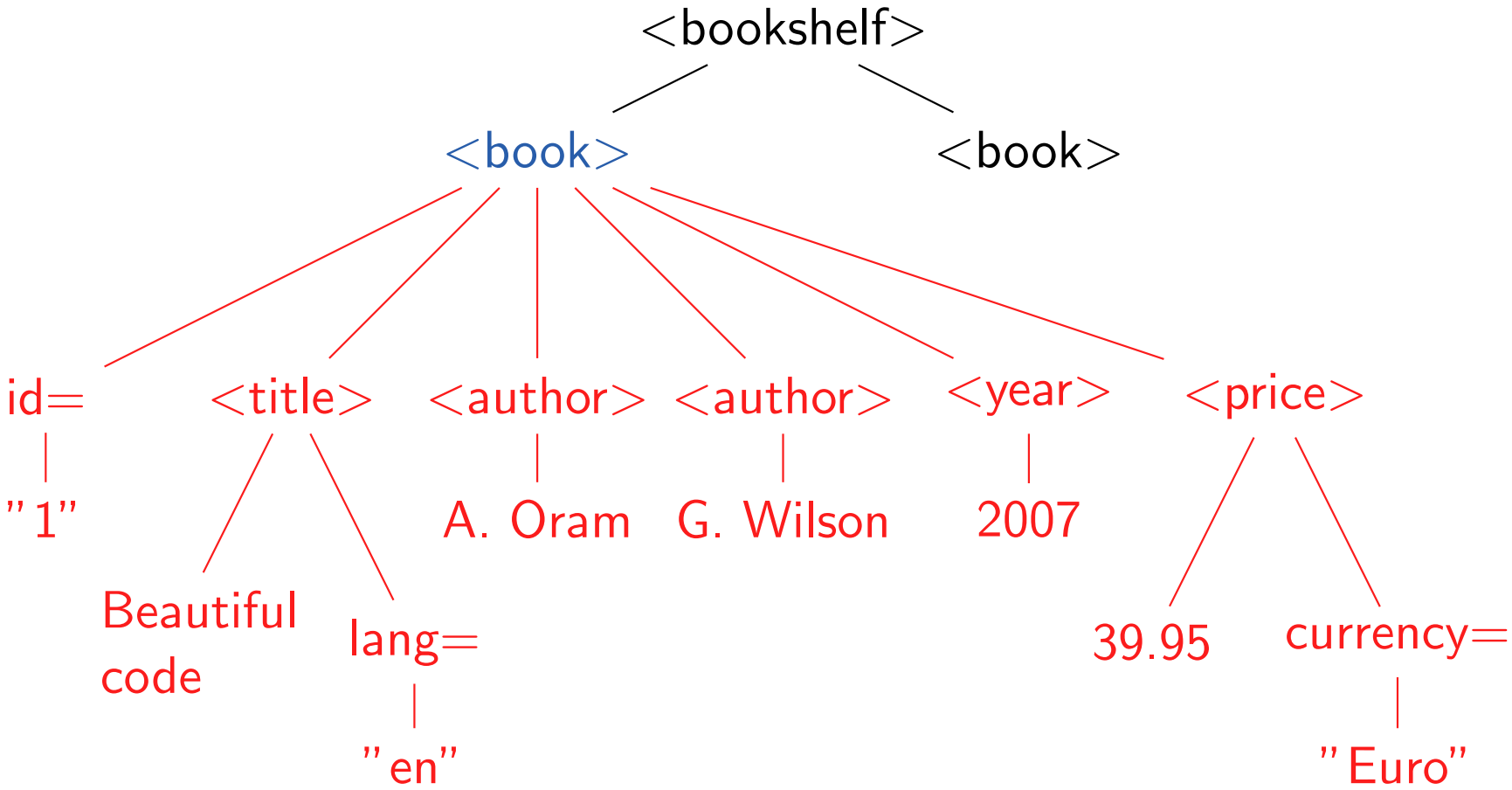
Children II


<bookshelf>
<book>
<book>
id=
"1"
<title>
Beautiful
code
lang=
"en"
<author>
A. Oram
<author>
G. Wilson
<year>
2007
<price>
39.95
currency=
"Euro"

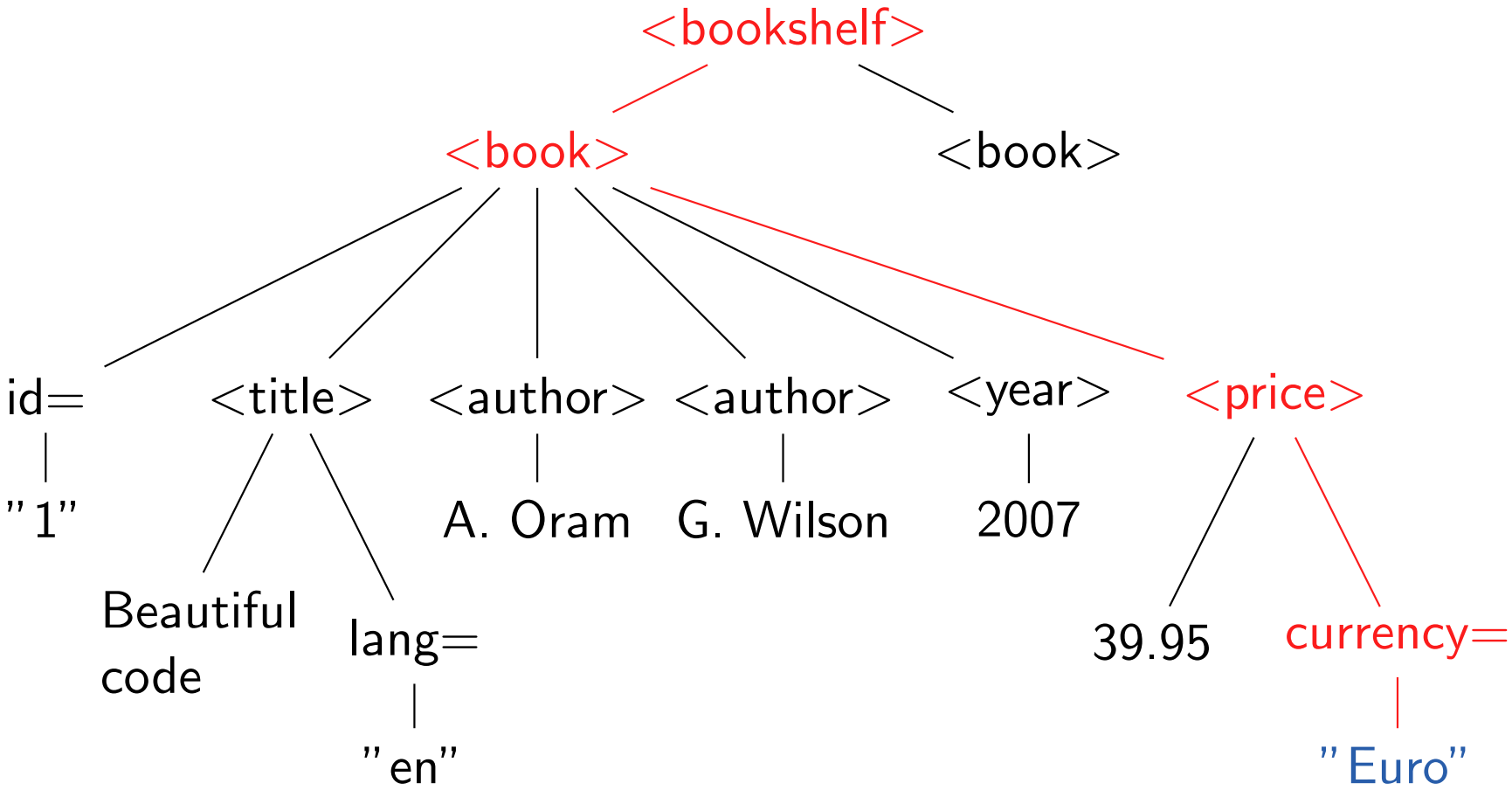
Parent


<bookshelf>
<book>
<book>
id=
"1"
<title>
Beautiful
code
lang=
"en"
<author>
A. Oram
<author>
G. Wilson
<year>
2007
<price>
39.95
currency=
"Euro"

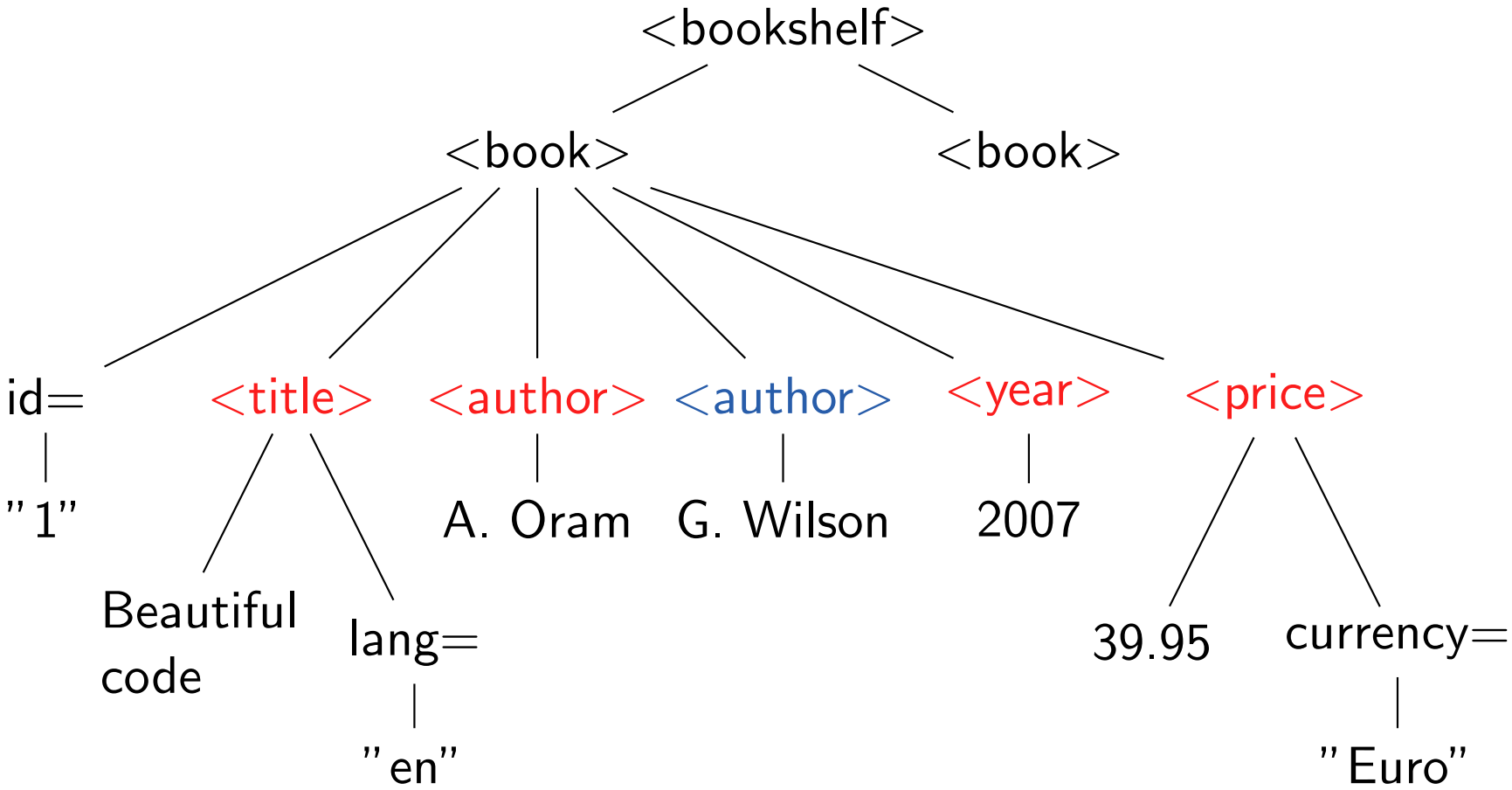
Descendants



Ascendants


<bookshelf>
<book>
<book>
id=
"1"
<title>
Beautiful
code
lang=
"en"
<author>
A. Oram
<author>
G. Wilson
<year>
2007
<price>
39.95
currency=
"Euro"

Siblings



Attribute siblings

Attributes are not considered siblings to elements.

CDATA

Avoid the escaping hell in text content.

Without CDATA

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>

    <hint>
      Some examples make use of &lt;xml&gt;.
    </hint>

  </book>
</bookshelf>
```

CDATA

Avoid the escaping hell in text content.

With CDATA

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>

    <hint>
      <![CDATA[
      Some examples make use of <xml>.
      ]]>
    </hint>

  </book>
</bookshelf>
```

CDATA

Avoid the escaping hell in text content.

The CDATA dilemma

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>

    <hint>
      <![CDATA[
      Some examples show the usage of <![CDATA[ ]]>
      ]]>
    </hint>

  </book>
</bookshelf>
```

CDATA

Avoid the escaping hell in text content.

The CDATA dilemma workaround

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>

    <hint>
      <![CDATA[
      Some examples show the usage of <![CDATA[ ]]]><![CDATA[>
      ]]>
    </hint>

  </book>
</bookshelf>
```

CDATA

Avoid the escaping hell in text content.

The CDATA dilemma solution

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>

    <hint>
U29tZSBleGFtcGxlcyBzaG93IHRoZSB1c2FnZSBvZiA8IVtDREFUQVsgXV0+
    </hint>

  </book>
</bookshelf>
```

Base 64 in PHP

Use the built in functions base64_encode() and base64_decode().

Comments

Comments in XML

```
<bookshelf>

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <!-- ... more books ... -->

</bookshelf>
```

Namespaces

Namespaces

Allow to avoid naming conflicts between different XML sources.

Single, default namespace

```
<bookshelf xmlns="http://example.com/book">

  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

</bookshelf>
```

Namespaces

Namespaces

Allow to avoid naming conflicts between different XML sources.

Multiple namespaces

```
<bookshelf
      xmlns="http://example.com/book"
      xmlns:book="http://example.com/book"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
>

   <book id="1">
      <dc:title book:lang="en">Beautiful code</dc:title>
      <author>A. Oram</author>
      <author>G. Wilson</author>
      <year>2007</year>
      <price currency="Euro">35.95</price>
   </book>

</bookshelf>
```

Outline

XML APIs

PHP has quite some XML APIs.

- The most important are:
  - DOM
  - XMLReader / XMLWriter
  - SimpleXML
- Deprecated are:
  - DOM XML
  - XML Parser

Overview - DOM

- Document Object Model
- Standardized API to access XML tree
  - W3C recommendation
  - Level 1 in 1999
  - Currently: Level 3 (2004)
- Available in many languages
  - C
  - Java
  - Perl
  - Python
  - ...
- Represents XML nodes as objects
- Loads full XML tree into memory

Overview - XMLReader/-Writer

- Popular approach to access XML data
- Similar implementations available in
  - Java
  - C#
- Pull / push based
- Does not load XML fully into memory

Overview - SimpleXml

- Very simple access to XML data
- Unique (?) to PHP
- Represents XML structures as objects
- Initial implementation hackish
- Loads full XML tree into memory
- You don't want to use SimpleXML, seriously!

APIs compared

	DOM	XMLReader/-Writer	SimpleXML
Read	⋆	⋆	●
Write	⋆	⋆	○
Manipulate	⋆	●	●
Full control	⋆	⋆	-
Namespaces	⋆	⋆	○
XPath	⋆	-	⋆
Validate	DTD Schema RelaxNG	DTD Schema RelaxNG	-
Comfort	●	○	⋆

- ⋆ Fully supported
- ● Supported but not nice
- ○ Poorly supported
- \- Not supported at all

Outline - XML in PHP

Introductional example

Printing all authors

```php
$dom = new DOMDocument();
$dom->load( 'sources/example.xml' );

$authors = $dom->getElementsByTagName( 'author' );

foreach ( $authors as $author )
{
    echo 'Author: ' . $author->nodeValue . "\n";
}
```

Output

```
Author: A. Oram
Author: G. Wilson
Author: T. Schlitt
Author: K. Nordmann
```

Essential classes

DOMNode

Purpose

Base class for all nodes types (elements, attributes, ...).

- Typical tree operations
  - `appendChild()`
  - `removeChild()`
  - `replaceChild()`
  - `hasChildNodes()`
  - `insertBefore()`

DOMNode

Purpose

Base class for all nodes types (elements, attributes, ...).

- Typical tree operations
  - `appendChild()`
  - `removeChild()`
  - `replaceChild()`
  - `hasChildNodes()`
  - `insertBefore()`
- DOM specific operations
  - `cloneNode()`
  - `lookupNamespaceURI()`
  - `lookupPrefix()`
  - `isDefaultNamespace()`
  - `normalize()`

DOMNode

Purpose

Base class for all nodes types (elements, attributes, ...).

- Typical tree operations
  - `appendChild()`
  - `removeChild()`
  - `replaceChild()`
  - `hasChildNodes()`
  - `insertBefore()`
- DOM specific operations
  - `cloneNode()`
  - `lookupNamespaceURI()`
  - `lookupPrefix()`
  - `isDefaultNamespace()`
  - `normalize()`

- Typical tree properties
  - `$parent`
  - `$childNodes`
  - `$previousSibling`
  - `$nextSibling`

DOMNode

Purpose

Base class for all nodes types (elements, attributes, ...).

- Typical tree operations
  - `appendChild()`
  - `removeChild()`
  - `replaceChild()`
  - `hasChildNodes()`
  - `insertBefore()`
- DOM specific operations
  - `cloneNode()`
  - `lookupNamespaceURI()`
  - `lookupPrefix()`
  - `isDefaultNamespace()`
  - `normalize()`
- Typical tree properties
  - `$parent`
  - `$childNodes`
  - `$previousSibling`
  - `$nextSibling`
- DOM specific properties
  - `$nodeType`
  - `$nodeValue`
  - `$ownerDocument`
  - `$namespaceURI`
  - `$prefix`
  - `$localName`

DOMElement

Purpose

Representation of an element (extends node)

- Attribute related operations
  - `hasAttribute[NS]()`
  - `getAttribute[NS]()`
  - `getAttributeNode[NS]()`
  - `setAttribute[NS]()`
  - `removeAttribute[NS]()`

DOMElement

Purpose

Representation of an element (extends node)

- Attribute related operations
  - `hasAttribute[NS]()`
  - `getAttribute[NS]()`
  - `getAttributeNode[NS]()`
  - `setAttribute[NS]()`
  - `removeAttribute[NS]()`
- Element related operations
  - `getElementsByTagName[NS]()`
  - `appendChild()` (inherited)
  - `removeChild()` (inherited)
  - `replaceChild()` (inherited)

DOMElement

Purpose

Representation of an element (extends node)

- Attribute related operations
  - `hasAttribute[NS]()`
  - `getAttribute[NS]()`
  - `getAttributeNode[NS]()`
  - `setAttribute[NS]()`
  - `removeAttribute[NS]()`
- Element related operations
  - `getElementsByTagName[NS]()`
  - `appendChild()` (inherited)
  - `removeChild()` (inherited)
  - `replaceChild()` (inherited)

- Properties
  - `$tagName`

Purpose

Representation of a XML document

- Creation methods
  - `createAttribute[NS]()`
  - `createElement[NS]()`

DOMDocument

Purpose

Representation of a XML document

- Creation methods
  - `createAttribute[NS]()`
  - `createElement[NS]()`
- Element retrieval methods
  - `getElementsByTagName[NS]()`
  - `getElementById()`

DOMDocument

Purpose

Representation of a XML document

- Creation methods
  - `createAttribute[NS]()`
  - `createElement[NS]()`
- Element retrieval methods
  - `getElementsByTagName[NS]()`
  - `getElementById()`
- Misc operations
  - `registerNodeClass()`
  - `validate()`
  - `schemaValidate()`
  - `relaxNGValidate()`

DOMDocument

Purpose

Representation of a XML document

- Creation methods
  - `createAttribute[NS]()`
  - `createElement[NS]()`
- Element retrieval methods
  - `getElementsByTagName[NS]()`
  - `getElementById()`
- Misc operations
  - `registerNodeClass()`
  - `validate()`
  - `schemaValidate()`
  - `relaxNGValidate()`
- Load /save methods
  - `load/save()`
  - `load/saveHTMLFile()`

DOMDocument

Purpose

Representation of a XML document

- Creation methods
  - `createAttribute[NS]()`
  - `createElement[NS]()`
- Element retrieval methods
  - `getElementsByTagName[NS]()`
  - `getElementById()`
- Misc operations
  - `registerNodeClass()`
  - `validate()`
  - `schemaValidate()`
  - `relaxNGValidate()`
- Load /save methods
  - `load/save()`
  - `load/saveHTMLFile()`
- Properties
  - `$documentElement`
  - `$documentURI`
  - `$preserveWhitespace`
  - `$formatOutput`
  - `$doctype`
  - `$xmlVersion`
  - `$xmlEncoding`
  - `$recover` (not DOM!)

DOMNodeList

Purpose

Collection of DOMNodes (elements, attributes, ...). Iteratable.

- Operations
  - `item()`
- Properties
  - `$length`

Standard

```
for ( $i = 0; $i < $authors->length; ++$i )
{
    echo 'Author: '
        . $authors->item( $i )->nodeValue . "\n";
}
```

DOMNodeList

Purpose

Collection of DOMNodes (elements, attributes, ...). Iteratable.

- Operations
  - `item()`
- Properties
  - `$length`

Standard

```
for ( $i = 0; $i < $authors->length; ++$i )
{
    echo 'Author: '
        . $authors->item( $i )->nodeValue . "\n";
}
```

Foreach

```
foreach ( $authors as $author )
{
    echo 'Author: '
        . $author->nodeValue . "\n";
}
```

Practical DOM

Reading XML I

Reading

```
$dom = new DOMDocument();
$dom->load( 'sources/example.xml' );

$root = $dom->documentElement;

echo 'Document is a ' . $root->tagName . "\n";

$books = $root->getElementsByTagName( 'book' );
```

Reading II

```
foreach ( $books as $book )
{
    echo ' Book with ID '
        . $book->getAttribute( 'id' );

    foreach ( $book->getElementsByTagName( 'price' ) as
        $price )
    {
        echo ' costs '
            . $price->nodeValue . ' '
            . $price->getAttribute( 'currency' )
            . "\n";
    }
}
```

Reading XML III

Output

```
Document is a bookshelf
 Book with ID 1 costs 35.95 Euro
 Book with ID 2 costs 39.95 Euro
```

Create XML I

Creating

```
$dom = new DOMDocument( '1.0', 'UTF-8' );
$dom->formatOutput = true;

$root = $dom->appendChild(
    $dom->createElement( 'dvdshelf' )
);

$dvd = $root->appendChild(
    $dom->createElement( 'dvd' )
);
```

Create XML II

Creating II

```
$dvd->setAttribute( 'id', 1 );

$dvd->appendChild(
    $dom->createElement( 'title', 'Star Trek' )
);

$dom->save( 'sources/dom_create.xml' );
```

Output

```
<?xml version="1.0" encoding="UTF-8"?>
<dvdshelf>
  <dvd id="1">
    <title>Star Trek</title>
  </dvd>
</dvdshelf>
```

Manipulate XML I

Source XML

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>    <book id="1"><title lang="en">Beautiful code</
    title>
        <author>A. Oram</author> <author>G. Wilson</author>
 <year>2007</year>
    <price
currency="Euro">35.95</price> </book>
     </bookshelf>
```

Manipulate XML II

Manipulating

```
$dom = new DOMDocument();
$dom->formatOutput = true;

$dom->load(
      'sources/example_weirdformat.xml',
     LIBXML_NOBLANKS
);
```

Manipulating II

```
$root = $dom->documentElement;

$newBook = $root->appendChild(
    $dom->createElement( 'book' )
);

$newBook->setAttribute( 'id', 2 );

$newBook->appendChild(
    $dom->createElement(
        'title', 'eZ Components - Das Entwicklerhandbuch'
    )
);

$dom->save( 'sources/dom_manipulate.xml' );
```

Manipulate XML IV

Output

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>
  <book id="2">
    <title>eZ Components - Das Entwicklerhandbuch</title>
  </book>
</bookshelf>
```

Outline - XML in PHP

Reading XML I

Reading

```
$reader = new XMLReader();
$reader->open( 'sources/example.xml' );

while( $reader->read() )
{
    if ( $reader->nodeType !== XMLReader::ELEMENT )
    {
        continue;
    }
```

Reading XML II

Reading

```
    if ( $reader->localName === 'book' )
    {
        echo 'Book with ID '
            . $reader->getAttribute( 'id' );
    }

    if ( $reader->localName === 'price' )
    {
        echo ' costs '
            . $reader->readString() . ' '
            . $reader->getAttribute( 'currency' )
            . "\n";
    }
}
```

Reading XML III

Output

```
Book with ID 1 costs 35.95 Euro
Book with ID 2 costs 39.95 Euro
```

Create XML I

Creating

```
$writer = new XMLWriter();
$writer->openUri( 'sources/xmlwriter_create.xml' );

$writer->setIndentString( '  ' );
$writer->setIndent( true );

$writer->startDocument( '1.0', 'UTF-8' );
$writer->startElement( 'dvdshelf' );
```

Create XML II

Creating II

```
$writer->startElement( 'dvd' );
$writer->writeAttribute( 'id', '1' );

$writer->writeElement( 'title', 'Star Trek' );

$writer->endElement(); // <dvd>

$writer->flush();

$writer->endElement(); // <dvdshelf>
$writer->endDocument();
```

Create XML III

Output

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>
  <book id="1">
    <title>Beautiful code</title>
  </book>
</bookshelf>
```

Outline - XML in PHP

Reading XML I

Reading

```php
$xml = simplexml_load_file( 'sources/example.xml' );

foreach ( $xml->book as $book )
{
    echo ' Book with ID '
         .  $book['id']
         . ' costs '
         . $book->price . ' '
         . $book->price['currency']
         . "\n";
}
```

Reading XML II

Output

```
Book with ID 1 costs 35.95 Euro
Book with ID 2 costs 39.95 Euro
```

Create XML I

Creating

```
$xml = new SimpleXmlElement( '<dvdshelf></dvdshelf>');

$xml->addChild( 'dvd' );
$xml->dvd[0]->addAttribute( 'id', 1 );

$xml->dvd[0]->addChild( 'title', 'Star Trek' );

$xml->asXML( 'sources/simplexml_create.xml' );
```

Output

```
<?xml version="1.0"?>
<dvdshelf><dvd id="1"><title>Star Trek</title></dvd></
    dvdshelf>
```

Outline

Outline - XPath

Overview

XPath

Enables you to select information parts from XML documents.

- Traverse the XML tree
- Select XML nodes
- W3C recommendation
- Version 1: November 1999
- Version 2: January 2007
- Fields of application
  - XSLT (XML Stylesheet Language Transformations)
  - Fetching XML nodes within programming languages

Overview

XPath

Enables you to select information parts from XML documents.

- Traverse the XML tree
- Select XML nodes
- W3C recommendation
- Version 1: November 1999
- Version 2: January 2007
- Fields of application
  - XSLT (XML Stylesheet Language Transformations)
  - Fetching XML nodes within programming languages

XML example reminder

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components – Das Entwicklerhandbuch</
        title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>
</bookshelf>
```

Introductory example

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components – Das Entwicklerhandbuch</
        title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>
</bookshelf>
```

2 variants to fetch all books

```
/bookshelf/book
//book
```

Introductory example

```
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>

  <book id="2">
    <title lang="de">eZ Components – Das Entwicklerhandbuch</
        title>
    <author>T. Schlitt</author>
    <author>K. Nordmann</author>
    <year>2007</year>
    <price currency="Euro">39.95</price>
  </book>
</bookshelf>
```

2 variants to fetch all authors

```
/bookshelf/book/author
//author
```

Outline - XPath

Addressing

- Every XPath expression matches a set of nodes (0..n)
- It encodes an "address" for the selected nodes
- Simple XPath expressions look similar to Unix file system addresses
- Two generally different ways of addressing are supported

Absolute addressing

```
/bookshelf/book/title
/bookshelf/book/author
```

Relative addressing

```
book/author
../title
```

Contexts

- Every expression step creates a new context
- The next is evaluated in the context created by the previous one

Contexts

```
/bookshelf/book/title
```

`/`	resets the context to global
`bookshelf`	selects all <bookshelf> elements in the global context
`/`	creates a new context, all children of <bookshelf>
`book`	selects all <book> elements in this context
`/`	creates a new context, all children of selected <book>s
`title`	selects all <title> elements in this context

→ A set of all title element nodes.

Attributes

Select attributes

Prepend the attribute name with an `@`.

Select all currency attributes

```
/bookshelf/book/price/@currency
```

Steps

Navigation

Navigation is not only possible in parent → child direction.

Navigate to parent

```
../
/bookshelf/book/title/../author
```

Navigate to descendants

```
//title
/bookshelf/book//@currency
```

Indexing

Access nodes by position

It is possible to access a specific node in a set by its position.

Indexing

```
/bookshelf/book[2]
/bookshelf/book/author[1]
```

Indexing

Access nodes by position

It is possible to access a specific node in a set by its position.

Indexing

```
/bookshelf/book[2]
/bookshelf/book/author[1]
```

Start index

- Indexing generally **1** based
- Some Internet Explorer versions start with **0**

Wildcards

Wildcard search

A wildcard represents a node of a certain type with arbitrary name.

Wildcards

```
/bookshelf/*/title
/bookshelf/book/@*
```

Union

Union

Union the node sets selected by multiple XPath expressions.

Union

```
/bookshelf/book/title | /bookshelf/book/author
```

A first PHP example

Querying first book title

```
$dom = new DOMDocument();
$dom->load( 'sources/example.xml' );

$xpath = new DOMXPath( $dom );

$titles = $xpath->query( '/bookshelf/book[1]/title' );

echo 'Title of first book is '
    . $titles->item( 0 )->nodeValue . "\n";
```

Output

```
Title of first book is Beautiful code
```

Second PHP example

Querying all currencies

```
// ...

$currencies = $xpath->query( '//price/@currency' );

echo "Following currencies occur:\n";

foreach ( $currencies as $currency )
{
    echo $currency->nodeValue . "\n";
}
```

Output

```
Following currencies occur:
Euro
Euro
```

Outline - XPath

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

You already saw examples

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

- 4 axis:
  - child (default)
  - attribute (`@`)
  - descendant-or-self (`//`)
  - parent (`..`)

You already saw examples

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

You already saw examples

- 4 axis:
  - child (default)
  - attribute (`@`)
  - descendant-or-self (`//`)
  - parent (`..`)
- Node tests:
  - name of the node (default)
  - wildcard (`*`)

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

You already saw examples

- 4 axis:
  - child (default)
  - attribute (`@`)
  - descendant-or-self (`//`)
  - parent (`..`)
- Node tests:
  - name of the node (default)
  - wildcard (`*`)
- Predicate:
  - accessing a node by index

XPath syntax

- An XPath query consists of steps
- Each step consists of:
  1. an axis
  2. a node test
  3. a predicate

You already saw examples

- 4 axis:
  - child (default)
  - attribute (@)
  - descendant-or-self (//)
  - parent (..)
- Node tests:
  - name of the node (default)
  - wildcard (*)
- Predicate:
  - accessing a node by index

Step syntax

```
<axis>::<nodetest>[<predicate>]
```

Axis

Axis

13 dimensions

Imagine an XML document to be a 13 dimensional space...

Axis	Shortcut
ancestor	
ancestor-or-self	
attribute	@
child	-
descendant	
descendant-or-self	//
following	
following-sibling	
namespace	
parent	..
preceding	
preceding-sibling	
self	.

Axis

Axis syntax

```
<axisname>::<nodetest>
```

Axis

Axis syntax

```
<axisname>::<nodetest>
```

Child axis

```
/bookshelf/book
child::bookshelf/child::book
```

Axis

Axis syntax

```
<axisname>::<nodetest>
```

Descendant-or-self axis

```
//book
descendant-or-self::book
```

Axis

Axis syntax

```
<axisname>::<nodetest>
```

Attribute axis

```
//book/@id
descendant-or-self::book/attribute::id
```

Axis

Axis syntax

```
<axisname>::<nodetest>
```

Parent axis

```
//book/@id/..
descendant-or-self::book/attribute::id/parent::node()
```

Ancestor axis

XPath with ancestor axis

```
//title/ancestor::*
```

Ancestor axis

XPath with ancestor axis

```
//title/ancestor::*
```

Selected XML nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <author>A. Oram</author>
    <author>G. Wilson</author>
    <year>2007</year>
    <price currency="Euro">35.95</price>
  </book>
  <!-- ... -->
</bookshelf>
```

Following axis

XPath with ancestor axis

```
//book/following::*
```

Following axis

XPath with ancestor axis

```
//book/following::*
```

Selected XML nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <!-- ... -->
    </book>

  <book id="2">
    <title lang="de">eZ Components - Das Entwicklerhandbuch</title>
    <!- ... ->
  </book>
</bookshelf>
```

Following-sibling axis

XPath with ancestor axis

```
//book/following-sibling::*
```

Following-sibling axis

XPath with ancestor axis

```
//book/following-sibling::*
```

Selected XML nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshelf>
  <book id="1">
    <title lang="en">Beautiful code</title>
    <!-- ... -->
    </book>

  <book id="2">
      <title lang="de">eZ Components - Das Entwicklerhandbuch
          </title>
      <!-- ... -->
  </book>
</bookshelf>
```

Namespace axis

XPath with namespace axis

```
namespace::*
```

Namespace axis

XPath with namespace axis

```
namespace::*
```

XML with namespaces

```
<bookshelf
      xmlns="http://example.com/book"
      xmlns:book="http://example.com/book"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
>
   <book id="1">
      <dc:title book:lang="en">Beautiful code</dc:title>
      <author>A. Oram</author>
      <author>G. Wilson</author>
      <year>2007</year>
      <price currency="Euro">35.95</price>
   </book>
</bookshelf>
```

Predicates

Predicate syntax

- You already saw indexing with numeric predicates
- Predicates can also be booleans
- Functions and operators allow fine grained tests

Predicate syntax

```
<nodetest>[<predicate>]
```

Predicate examples

Select all books with ID 1

```
//book[@id = '1']
```

Predicate examples

Select all books that have any attribute at all

```
//book[@*]
//book/@*/..
```

Predicate examples

Select all books with price round 40

```
//book[round( price ) = 40]
```

Predicate examples

Select all authors with first name initial T

```
//book/author[substring(., 1, 1) = 'T']
```

Operator overview

Mathematical operators

$+$, $-$, $*$	Addition, subtraction, multiplication
div	Division
mod	Modulo operation

Operator overview

Mathematical operators

$+$, $-$, $*$	Addition, subtraction, multiplication
div	Division
mod	Modulo operation

Comparison operators

$=$	Check for equality
$!=$	Check for inequality
$<$, $<=$	Less than and less than or equal
$>$, $>=$	Greater than and greater than or equal

Operator overview

Mathematical operators

$+, -, *$	Addition, subtraction, multiplication
div	Division
mod	Modulo operation

Comparison operators

$=$	Check for equality
$!=$	Check for inequality
$<, <=$	Less than and less than or equal
$>, >=$	Greater than and greater than or equal

Logical operators

or	Logical or
and	Logical and

Operator overview

Mathematical operators

+, −, ∗	Addition, subtraction, multiplication
div	Division
mod	Modulo operation

Comparison operators

=	Check for equality
! =	Check for inequality
<, <=	Less than and less than or equal
>, >=	Greater than and greater than or equal

Logical operators

or	Logical or
and	Logical and

Logical negation

not() is a function in XPath!

Functions by example

- String functions
  - string-join() Concatenates 2 strings
  - substring() Extracts a part from a string

Functions by example

- String functions
  - string-join() Concatenates 2 strings
  - substring() Extracts a part from a string
- Node set functions
  - count() Returns number of nodes in a set
  - position() Returns the position index of each node

Functions by example

- String functions
  - string-join() Concatenates 2 strings
  - substring() Extracts a part from a string
- Node set functions
  - count() Returns number of nodes in a set
  - position() Returns the position index of each node
- Boolean functions
  - not() Negates the received boolean expression
  - true() Boolean true

Functions by example

- String functions
  - string-join() Concatenates 2 strings
  - substring() Extracts a part from a string
- Node set functions
  - count() Returns number of nodes in a set
  - position() Returns the position index of each node
- Boolean functions
  - not() Negates the received boolean expression
  - true() Boolean true
- Mathematical functions
  - round() Rounds the given number to the next integer
  - floor() Returns the next integer smaller than the given number

Functions by example

- String functions
  - string-join() Concatenates 2 strings
  - substring() Extracts a part from a string
- Node set functions
  - count() Returns number of nodes in a set
  - position() Returns the position index of each node
- Boolean functions
  - not() Negates the received boolean expression
  - true() Boolean true
- Mathematical functions
  - round() Rounds the given number to the next integer
  - floor() Returns the next integer smaller than the given number

Function overview

An overview on all functions can be found on
`http://www.w3.org/TR/xpath-functions/`

Outline - XPath

Outline - XPath in action

Namespaces in RDF

Find all namespaces in an RDF document

RDF Resource Description Framework

- Semantic web
- Makes heavy usage of different namespaces

Complex variant

```
//*[ name(.) = 'rdf:Description'
]/*[ namespace-uri(.) != namespace-uri(..) and
     namespace-uri(.) != '' and
     namespace-uri(.) != namespace-uri(preceding-
         sibling::*)]
```

Find all namespaces in an RDF document

RDF Resource Description Framework

- Semantic web
- Makes heavy usage of different namespaces

Complex variant

```
//*[ name(.) = 'rdf:Description'
]/*[ namespace-uri(.) != namespace-uri(..) and
     namespace-uri(.) != '' and
     namespace-uri(.) != namespace-uri(preceding-
         sibling::*)]
```

Simpler variant

```
//*[ name(.) = 'rdf:Description']/namespaces::*
```

Outline - XPath in action

pQuery

jQuery

- Cool JavaScript framework
- Allows selecting HTML elements by CSS selectors
- `http://jquery.com/`

pQuery

- Little example class
- Models a tiny bits of jQuery, using
  - DOM
  - XPath

Example HTML

```
<html>
  <head>
    <title>Some website</title>
  </head>
<body>
  <h1>Some headline</h1>
  <p>
    Some nice <a href="test.html" class="internal">content</a
        >.
  </p>
  <h1 id="second">Second headline</h1>
  <p>
    More nice <a href="test.html">content</a>.
  </p>
</body>
</html>
```

Usage

```php
require 'pquery/pquery.php';

$dom = new DOMDocument();
$dom->formatOutput = true;
$dom->loadHTMLFile( '../example.html' );

$q = new pQuery( $dom );

$q->query( 'h1#second' );
$q->addClass( 'someclass' );

$dom->saveHTMLFile( '../pquery_simple.html' );
```

The pQuery class

```
class pQuery
{
    protected $dom;

    protected $context;

    protected $xpath;

    public function __construct( DOMDocument $dom )
    {
        $this->dom     = $dom;
        $this->context = array( $dom->documentElement );
        $this->xpath   = new DOMXPath( $dom );
    }

    // ...
```

Issueing a query

```
// ...

public function query( $string )
{
    $xpath = $this->createXPath( $string );

    echo "Querying with XPath '$xpath'.\n";

    $newContext = array();
    foreach ( $this->context as $node )
    {
        $nodeList = $this->xpath->query( $xpath, $node );
        foreach ( $nodeList as $newNode )
        {
            $newContext[] = $newNode;
        }
    }
    $this->context = $newContext;
}

// ...
```

Creating XPath

```
// ...

protected function createXPath( $string )
{
    $parts = explode( ' ', $string );

    $xpath = array();
    foreach ( $parts as $part )
    {
        $xpath[] = $this->createSingleXPath( $part );
    }

    return implode( ' | ', $xpath );
}

// ...
```

Creating XPath part I

```
// ...

protected function createSingleXPath( $string )
{
    $parts = preg_split(
        '((#|\.))',
        $string,
        -1,
        PREG_SPLIT_DELIM_CAPTURE
    );

    $xpath = '//';

    // ...
```

Creating XPath part II

```
        // ...

        if ( count( $parts ) < 3 )
        {
            $xpath .= $string;
        }
        else
        {
            $xpath .= $this->createSelector( $parts );
        }

        return $xpath;
    }

    // ...
```

Creating selector

```
    // ...

    protected function createSelector( array $parts )
    {
        switch ( $parts[1] )
        {
            case '.':
                return $parts[0]
                    . '[contains( @class , " ' . $parts[2] . ' " )
                        ]';
                break;

            case '#':
                return $parts[0]
                    . '[@id = " ' . $parts[2] . ' "]';
                break;
        }
    }
}
```

Adding a class I

```
// ...

public function addClass( $class )
{
    foreach ( $this->context as $node )
    {
        if ( $node->nodeType !== XML_ELEMENT_NODE )
        {
            continue;
        }

        if ( !$node->hasAttribute( 'class' ) )
        {
            $classes = array();
        }
        else
        {
            $classes = explode(
                ' ',
                $node->getAttribute( 'class' )
            );
        }

        // ...
```

Adding a class II

```
            // ...

            if ( !in_array( $class, $classes ) )
            {
                $classes[] = $class;
            }

            $node->setAttribute(
                'class',
                implode( ' ', $classes )
            );
        }
    }

    // ...
```

Simple example

```php
require 'pquery/pquery.php';

$dom = new DOMDocument();
$dom->formatOutput = true;
$dom->loadHTMLFile( '../example.html' );

$q = new pQuery( $dom );

$q->query( 'h1#second' );
$q->addClass( 'someclass' );

$dom->saveHTMLFile( '../pquery_simple.html' );
```

Simple example result

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
    "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=
    UTF-8">
<title>Some website</title>
</head>
<body>
  <h1>Some headline</h1>
  <p>
    Some nice <a href="test.html" class="internal">content</a
        >.
  </p>
  <h1 id="second" class="someclass">Second headline</h1>
  <p>
    More nice <a href="test.html">content</a>.
  </p>
</body>
</html>
```

Advanced example

```php
require 'pquery/pquery.php';

$dom = new DOMDocument();
$dom->formatOutput = true;
$dom->loadHTMLFile( '../example.html' );

$q = new pQuery( $dom );

$q->query( 'h1 a.internal' );
$q->addClass( 'coolclass' );

$dom->saveHTMLFile( '../pquery_advanced.html' );
```

Advanced example result

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
    "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=
    UTF-8">
<title>Some website</title>
</head>
<body>
  <h1 class="coolclass">Some headline</h1>
  <p>
    Some nice <a href="test.html" class="internal coolclass">
        content</a>.
  </p>
  <h1 id="second" class="coolclass">Second headline</h1>
  <p>
    More nice <a href="test.html">content</a>.
  </p>
</body>
</html>
```

The end

Thank you for listening!

The end

Thank you for listening!

- Are there any questions left?

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?
- Contact me: Tobias Schlitt <toby@php.net>

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?
- Contact me: Tobias Schlitt <toby@php.net>
- Enjoy the conference!