```
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

Structured Web Documents in XML (b)

Adapted from slides from Grigoris Antoniou and Frank van Harmelen

Outline

(1) Introduction

(2) XML details

(3) Structuring

- DTDs
- XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

(7) Summary

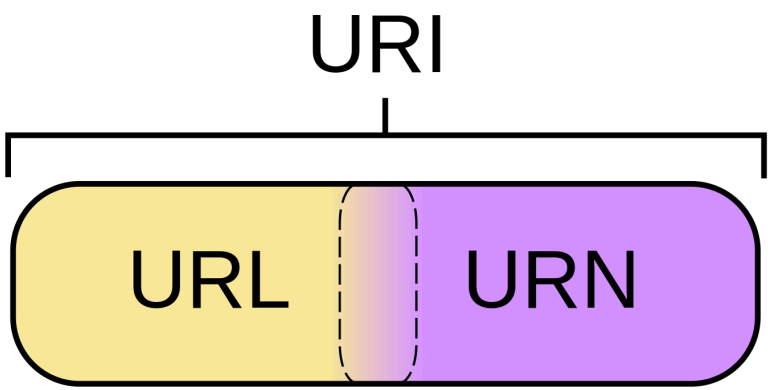
What's a Namespace?

- Wikipedia: "In computing, a **namespace** is a set of symbols that are used to organize objects of various kinds, so that these objects may be referred to by name"
- Useful when we need to combine or integrate multiple vocabularies
- Providing a way to avoid name collisions

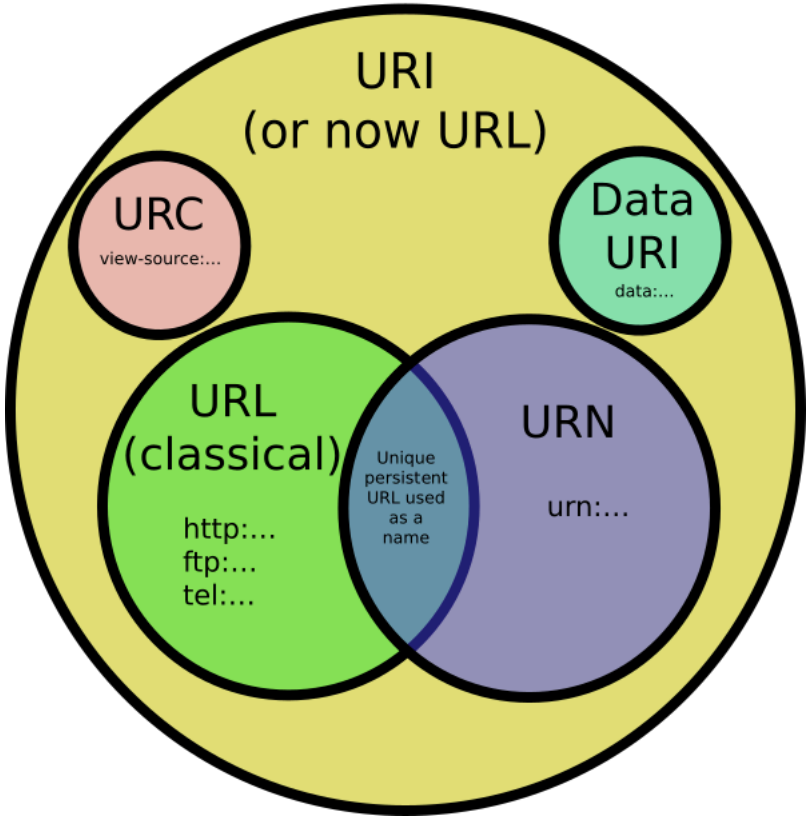
Namespaces

- XML namespaces provide uniquely named elements & attributes in an XML document
- XML document may use >1 DTD or schemas
- Since each was developed independently, **name collisions** can occur
- Solution: use different prefix
  prefix:name
- Where a prefix is associated with a URI, which can be a DTD or schema file
- **Namespaces even more important in RDF**

URLs, URNs, URIs, …



Venn diagram of URIs
as defined by the W3C



An Example

```
<vu:instructors
   xmlns:vu="http://www.vu.com/empDTD"
   xmlns:gu="http://www.gu.au/empDTD"
   xmlns:uky=http://www.uky.edu/empDTD
<uky:faculty uky:title="assistant professor"
   uky:name="John Smith"
   uky:department="Computer Science"/>
<gu:academicStaff  gu:title="lecturer"
   gu:name="Mate Jones"
   gu:school="Information Technology"/>
</vu:instructors>
```

Namespace Declarations

- Namespaces declared within elements for use in it and its children (elements and attributes)
- Namespace declaration has form:
  - **xmlns:prefix="location"**
  - **location** is a URI, often the DTD or schema file
- If no prefix specified: **xmlns="location"** then the **location** is used as the *default* prefix
- We'll see this same idea used in RDF

Outline

(1) Introduction

(2) Description of XML

(3) Structuring

- DTDs
- XML Schema

(4) Namespaces

(5) Accessing, querying XML docs: XPath

(6) Transformations: XSLT

Addressing & Querying XML Documents

- In relational databases, parts of a database can be selected and retrieved using SQL
  - Also very useful for XML documents
  - **Query languages**: XQuery, XQL, XML-QL
- The central concept of XML query languages is a **path expression**
  - Specifies how a node or set of nodes, in the tree representation, can be reached
- Useful for extracting data from XML

XPath

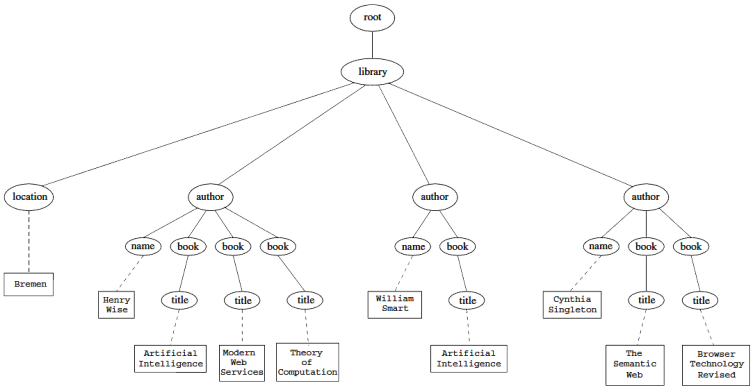
- XPath is core for XML query languages
- Gives a way to select nodes in XML documents
  - Operates on the tree data model of XML
  - Has a non-XML syntax
- Versions
  - XPath 1.0 (1999) is widely supported
  - XPath 2.0 (2007) more expressive subset of Xquery
  - XPath 3.1 (2017) current version, more features

Types of Path Expressions

- **Absolute** (starting at the root of the tree)
  - Syntactically they begin with the symbol **/**
  - It refers to the root of the document (one level above document's root element)
- **Relative** to a context node

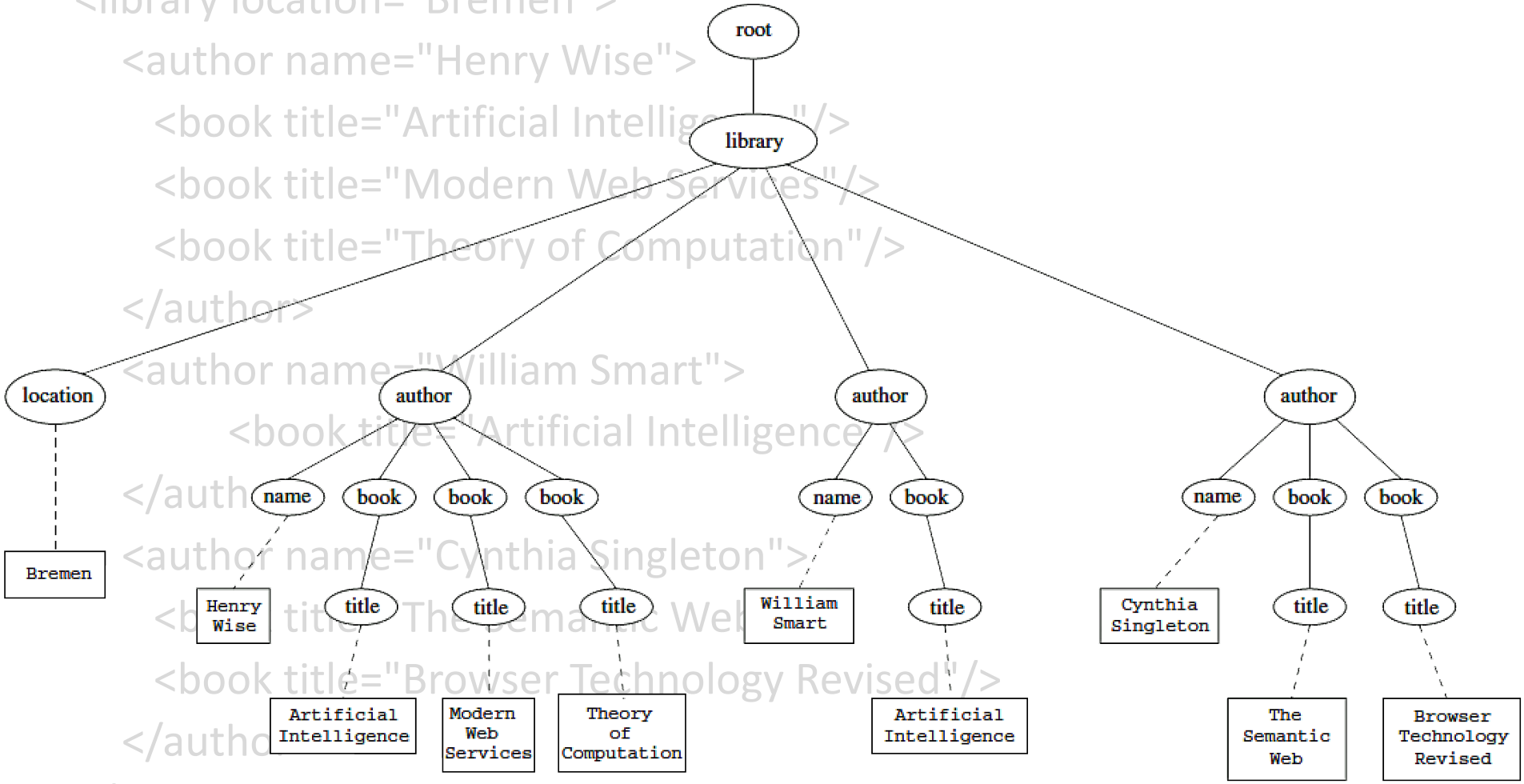
An XML Example

```
<library location="Bremen">
   <author name="Henry Wise">
     <book title="Artificial Intelligence"/>
     <book title="Modern Web Services"/>
     <book title="Theory of Computation"/>
   </author>
   <author name="William Smart">
         <book title="Artificial Intelligence"/>
   </author>
   <author name="Cynthia Singleton">
     <book title="The Semantic Web"/>
     <book title="Browser Technology Revised"/>
   </author>
</library>
```



Tree Representation

```
<library location="Bremen">
  <author name="Henry Wise">
    <book title="Artificial Intelligence"/>
    <book title="Modern Web Services"/>
    <book title="Theory of Computation"/>
  </author>
  <author name="William Smart">
    <book title="Artificial Intelligence"/>
  </author>
  <author name="Cynthia Singleton">
    <book title="The Semantic Web"/>
    <book title="Browser Technology Revised"/>
  </author>
</library>
```



Examples of Path Expressions in XPath

- **Q1: /library/author**
  - Addresses **all** **author** elements that are children of the **library** element node immediately below root
  - **/t1/…/tn**, where each **ti+1** is a child node of **ti**, is a path through the tree representation
- **Q2: //author**
  - Consider all elements in document and check whether they are of type **author**
  - Path expression addresses all **author** elements **anywhere** in the document

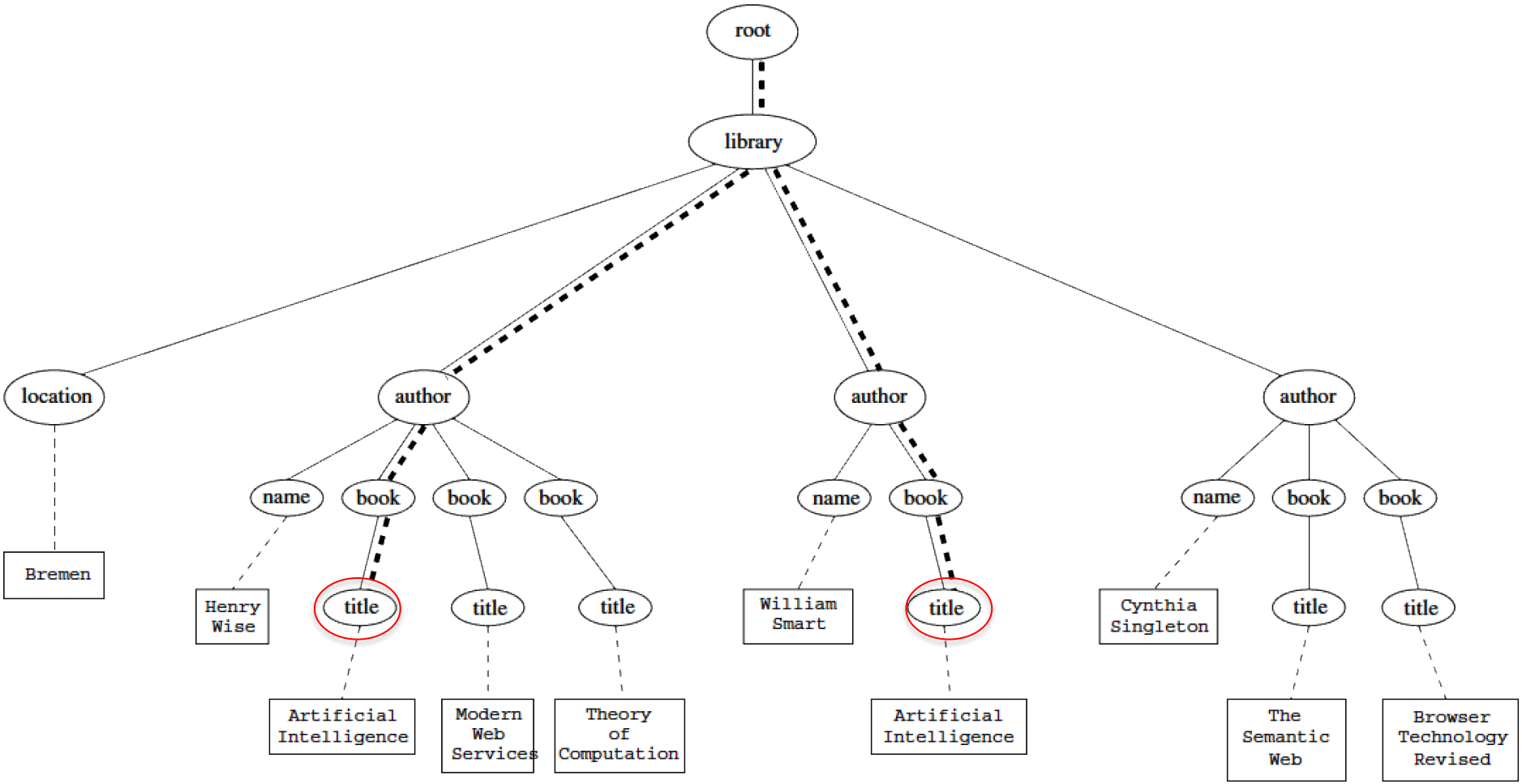
Examples of Path Expressions in XPath

- **Q3: /library/@location**
  - Addresses location attribute nodes within library element nodes
  - The symbol **@** is used to denote attribute nodes

- **Q4: //book/@title="Artificial Intelligence”**
  - Addresses all title attribute nodes within book elements anywhere in the document that have the value “Artificial Intelligence”

Tree Representation of Query 4

//book/@title="Artificial Intelligence”

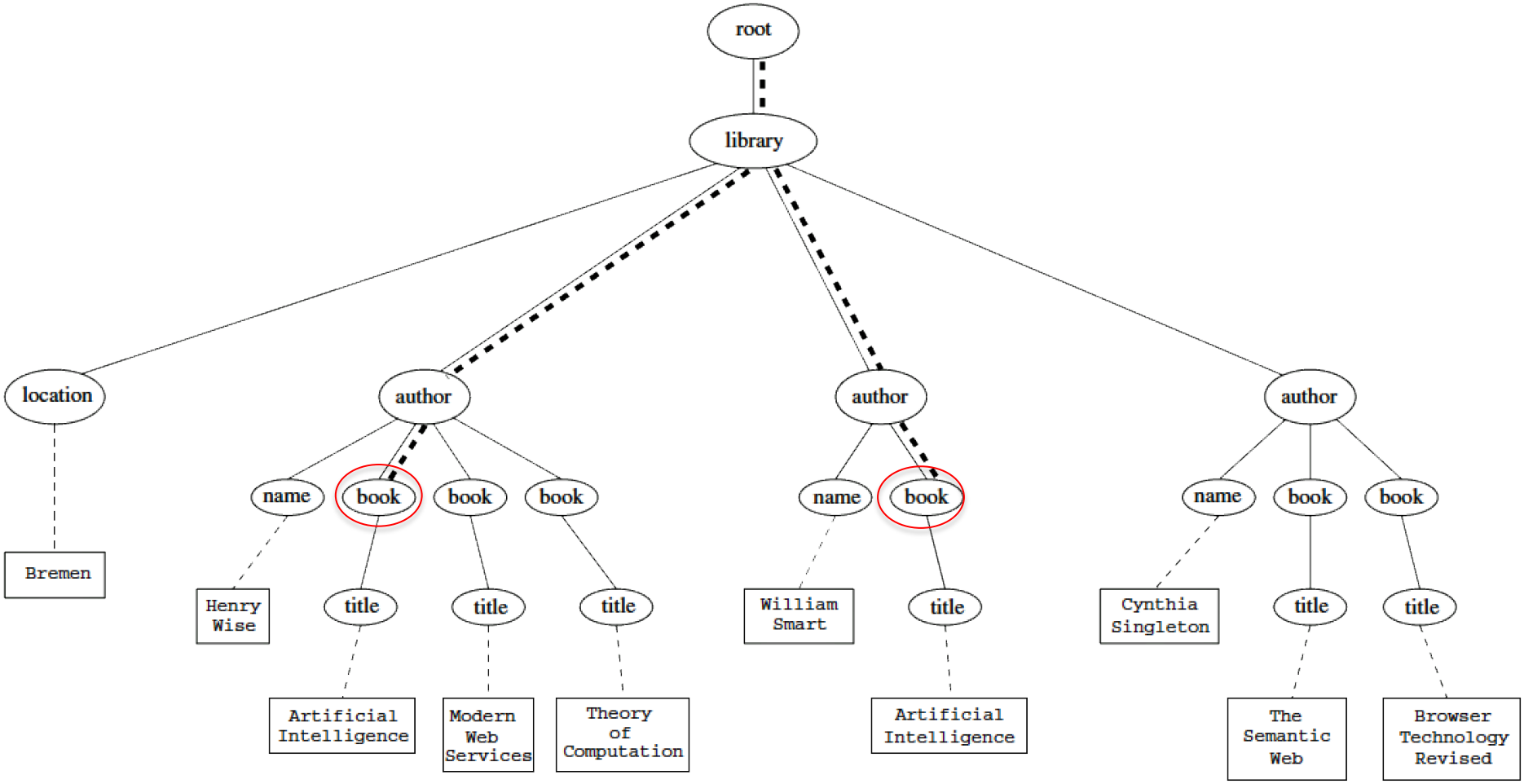


Examples of Path Expressions in XPath

- **Q5: /book[@title="Artificial Intelligence"]**
  - Addresses all books with title “Artificial Intelligence”
  - A test in brackets is a **filter expression** that restricts the set of addressed nodes.
  - Note differences between Q4 and Q5:
    - Query 5 addresses **book** elements, the **title** of which satisfies a certain condition.
    - Query 4 collects **title** attribute nodes of **book** elements

Tree Representation of Query 5

/book[@title="Artificial Intelligence"]



Examples of Path Expressions in XPath

- Q6: Address **first** author element node in the XML document

  //author[1]

- Q7: Address **last** book element within the first author element node in the document

  //author[1]/book[last()]

- Q8: Address all book element nodes **without a title** attribute

  //book[not @title]

Working with XML in Python

library

localhost:8889/notebooks/library.ipynb

jupyter library (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

The Python xml package has many tools for working with XML data and partial support for XPATH. There are other xml packages that are more powerful, but also more complex

In [1]:
```
import xml.etree.ElementTree as ET
```

read the simple book example used in the notes and SWP

In [2]:
```
library_data = ET.parse('library.xml')
```

```
What does its string representation look like?
```

In [3]:
```
root = library_data.getroot()
ET.tostring(root)
```

Out[3]:
```
b'<library location="Bremen">\n\t<author name="Henry Wise">\n\t   <book title="Artificial Int
elligence" />\n\t   <book title="Modern Web Services" />\n\t   <book title="Theory of Computa
tion" />\n\t</author>\n\t<author name="William Smart">\n\t\t<book title="Artificial Intellige
nce" />\n\t</author>\n\t<author name="Cynthia Singleton">\n\t   <book title="The Semantic We
b" />\n\t   <book title="Browser Technology Revised" />\n\t</author>\n</library>'
```

we can easily access its parts

In [4]:
```
print('ROOT:', root)
print('TAG:', root.tag)
print('ATTRIBUTES:', root.attrib)
print('ELEMENTS', [element for element in root])
```

```
ROOT: <Element 'library' at 0x104128ef8>
TAG: library
ATTRIBUTES: {'location': 'Bremen'}
ELEMENTS [<Element 'author' at 0x10417f818>, <Element 'author' at 0x10417f958>, <Element 'aut
hor' at 0x10417f9f8>]
```

Outline

(1) Introduction

(2) Description of XML

(3) Structuring

- DTDs
- XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

Displaying XML Documents

```
<author>
    <name>Grigoris Antoniou</name>
    <affiliation>University of Bremen</affiliation>
    <email>ga@tzi.de</email>
</author>
```

may be displayed in different ways:

Grigoris Antoniou
University of Bremen
ga@tzi.de

Grigoris Antoniou
University of Bremen
ga@tzi.de

Idea: use an external *style sheet* to transform an XML tree into an HTML or XML tree

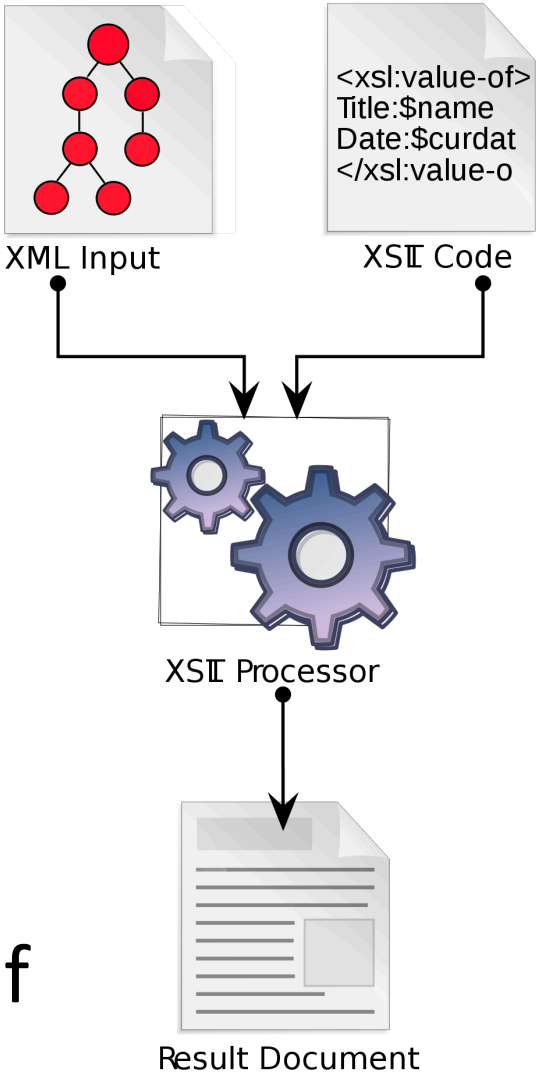
Style Sheets

- Style sheets can be written in various languages
  - E.g. CSS2 (cascading style sheets level 2)
  - XSL (extensible stylesheet language)
- XSL includes
  - a transformation language (XSLT)

    XSLT 3.0 is the current spec as of 2017
  - a formatting language
  - Both are XML applications

XSL Transformations (XSLT)

- XSLT specifies rules to transform XML document to
  - another XML document
  - HTML document
  - plain text
- Output may use same DTD or schema, or completely different vocabulary
- XSLT can be used independently of formatting language



XSLT Use Cases

- Move data & metadata from one XML representation to another
- Share information between applications using different schemas
- Processing XML content for ingest into a program or database
- The following example show XSLT used to display XML documents as HTML

XSLT Transformation into HTML

```
<author>
  <name>Grigoris Antoniou</name>
  <affiliation>University of Bremen
     </affiliation>
  <email>ga@tzi.de</email>
</author>
```

```
<xsl:template match="/author">
  <html>
      <head><title>An author</title></head>
      <body bgcolor="white">
            <b><xsl:value-of select="name"/></b><br/>
            <xsl:value-of select="affiliation"/><br/>
            <i><xsl:value-of select="email"/></i>
      </body>
  </html>
</xsl:template>
```

Style Sheet Output

```
<author>
  <name>Grigoris Antoniou</name>
  <affiliation>University of Bremen</affiliation>
  <email>ga@tzi.de</email>
</author>
```

```
<xsl:template match="/author"> <html>
      <head><title>An author</title></head>
      <body bgcolor="white">
        <b><xsl:value-of select="name"/></b><br/>
        <xsl:value-of select="affiliation"/><br/>
        <i><xsl:value-of select="email"/></i>
      </body>
  </html></xsl:template>
```

```
<html>
   <head><title>An author</title></head>
   <body bgcolor="white">
         <b>Grigoris Antoniou</b><br/>
         University of Bremen<br/>
         <i>ga@tzi.de</i>
   </body>
</html>
```

Observations About XSLT

- XSLT documents are XML documents
  - XSLT sits on top of XML
- The XSLT document defines a **template**
  - In this case, an HTML document with placeholders for content to be inserted
- **xsl:value-of** retrieves value of an element and copies it into output document
  - It places some content into the template

Auxiliary Templates

- We may have XML documents with details of several authors
- It is a waste of effort to treat each **author** element separately
- In such cases, a special template is defined for **author** elements, which is used by the main template

Example of an Auxiliary Template

```
<authors>
  <author>
      <name>Grigoris Antoniou</name>
      <affiliation>University of Bremen</affiliation>
      <email>ga@tzi.de</email>
  </author>
  <author>
      <name>David Billington</name>
      <affiliation>Griffith University</affiliation>
      <email>david@gu.edu.net</email>
  </author>
</authors>
```

Example of an Auxiliary Template

```
<xsl:template match="/">
  <html>
    <head><title>Authors</title></head>
    <body bgcolor="white">
      <xsl:apply-templates select="author"/>
      <!-- apply templates for AUTHORS children -->
    </body>
  </html>
</xsl:template>
```

Example of an Auxiliary Template

```
<xsl:template match="authors">
    <xsl:apply-templates select="author"/>
</xsl:template>


<xsl:template match="author">
  <h2><xsl:value-of select="name"/></h2>
  <p> Affiliation:<xsl:value-of select="affiliation"/><br/>
  Email: <xsl:value-of select="email"/> </p>
</xsl:template>
```

Multiple Authors Output

```
<html>
  <head><title>Authors</title></head>
  <body bgcolor="white">
      <h2>Grigoris Antoniou</h2>
      <p>Affiliation: University of Bremen<br/>
      Email: ga@tzi.de</p>
      <h2>David Billington</h2>
      <p>Affiliation: Griffith University<br/>
      Email: david@gu.edu.net</p>
  </body>
</html>
```

How to apply XSLT transforms

- When modern browsers load an XML file, they apply a linked XSLT and display the results (hopefully HTML!)
- You can also explicitly use
  - An external Web service
  - An XML editor
  - A module/library of your favorite programming language

An XSLT Web Service

W3C XSLT Servlet

www.w3.org/2005/08/online_...

W3C Systems team

Online XSLT 2.0 Service

Important: W3C runs this service for its own use. The service,runs on Jigsaw, is based on Saxon and supports XSLT 2.0, is available publicly, but usage is subject to the conditions set forth below.

Inputs

URI for xsl resource:

URI for xml resource:

Attempt recursive authentication

Output

Forward language/content accept headers

Content-Type:

gzip compress output

Debug

Debug output

Show Trace

Suppress Transform output

Validate

transform

http://www.w3.org/2005/08/online_xslt/

CD Catalog example

```
<?xml-stylesheet type="text/xsl"
href="cdcatalog.xsl"?>
<catalog>
<cd>
  <title>Empire Burlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
<cd>
  <title>Hide your heart</title>
  <artist>Bonnie Tyler</artist>
  <country>UK</country>
  <company>CBS Records</company>
…
</cd> …
```

```
<xsl:template match="/">
 <html> <body>
 <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th align="left">Title</th>
    <th align="left">Artist</th>
   </tr>
   <xsl:for-each select="catalog/cd">
    <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
    </tr>
   </xsl:for-each>
  </table>
 </body> </html>
</xsl:template>
</xsl:stylesheet>
```

See these files online

Viewing an XML file in a Browser

curl –L http://bit.ly/CdCat19

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
<cd>
 <title>Empire Burlesque</title>
 <artist>Bob Dylan</artist>
 <country>USA</country>
 <company>Columbia</company>
 <price>10.90</price>
 <year>1985</year>
</cd>
<cd>
 <title>Hide your heart</title>
 <artist>Bonnie Tyler</artist>
 <country>UK</country>
 <company>CBS Records</company>
 <price>9.90</price>
 <year>1988</year>
</cd>
...
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown

XML Summary

- XML is a metalanguage that allows users to define markup
- XML separates content and structure from formatting
- XML is (one of the) the de facto standard to represent and exchange structured information on the Web
- XML is supported by query languages

Comments for Discussion

- Nesting of tags has *no standard meaning*
- *Semantics* of XML documents not accessible to machines and may or may not be for people
- Collaboration & exchange supported if there is underlying shared understanding of vocabulary
- XML well-suited for **close collaboration** where domain or community-based vocabularies are used; less so for global communication
- Databases went from tree structures (60s) to relations (80s) and graphs (10s)