XML and web services

Monday, March 14, 2011
2:50 PM

So far, we've discussed the use of XML in creating web services.

How does this work?

What other things can we do with it?

Where we're going…

Monday, March 29, 2010
2:50 PM

Where we're going…

- Business process can be described in XML…
- Which is defined by Xschemas…
- And utilizes queries called XPATHs…
- Which are both understood by Eclipse…
- And can be used to model business process and predict business value…

The main issue

Monday, April 11, 2011
4:31 PM

No program can be guaranteed to work on **all** inputs.
We need a "**guard clause**" that defines the inputs it works on.
XML has very powerful mechanism for

- Defining guards (Xschemas)
- Writing programs that use guards (XPATHs)

Basics of XML

Every XML document contains a single element

<foo>

…

</foo>

that can contain sub-elements.

Elements can have attributes, e.g.,

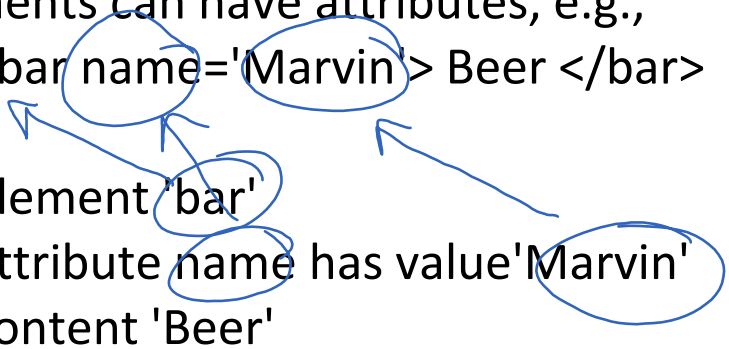
<bar name='Marvin'> Beer </bar>

has

element 'bar'

attribute name has value'Marvin'

content 'Beer'



XML and HTML: new rules:

All elements must be closed:

 =>
</br>

All attributes must be quoted strings (with " or ')

<ol type=a> => <ol type="a"> or <ol type='a'>

A shorthand

<cat/> means <cat></cat>

Monday, March 29, 2010
1:20 PM

Xschemas

- Generalization of Document Type Definitions (DTDs)
- Describe allowable contents of an XML element...
- … in XML itself!

Really strange:

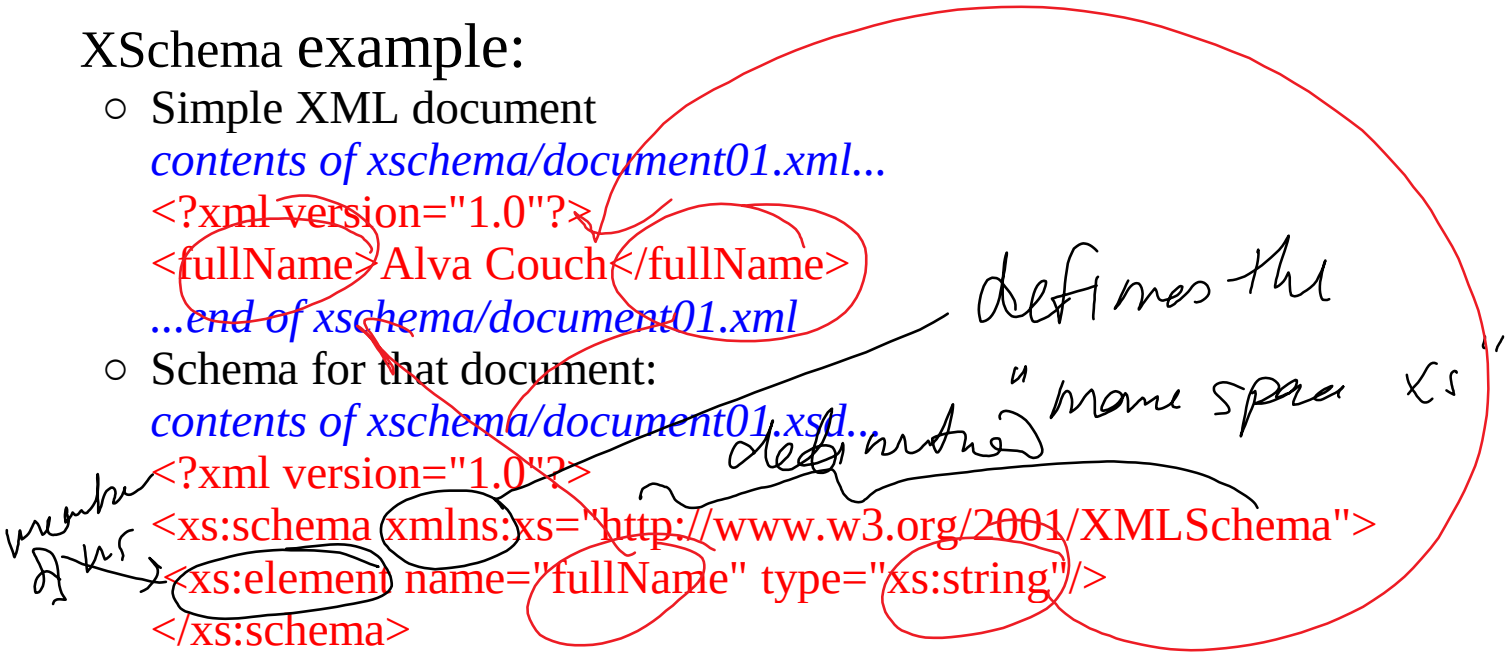
- Can think of an Xschema as a "type".
- A type in a programming language is a set of limitations.
- Likewise, an Xschema is a set of limitations for an XML document.

Xschema notation

Monday, March 29, 2010
1:28 PM

XSchema example:

- Simple XML document
  contents of xschema/document01.xml...
  ```
  <?xml version="1.0"?>
  <fullName>Alva Couch</fullName>
  ```
  ...end of xschema/document01.xml
- Schema for that document:
  contents of xschema/document01.xsd...
  ```
  <?xml version="1.0"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fullName" type="xs:string"/>
  </xs:schema>
  ```



- This says that the element "fullName" should contain text.
  Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Apologies in advance for self-plagiarism: I'm picking the important points from the old Comp150WEB.

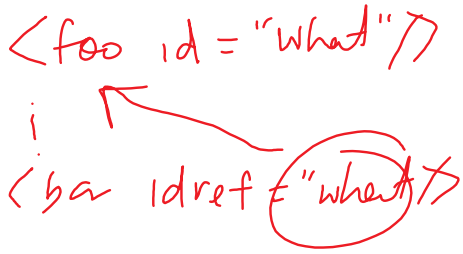
Kinds of Xschema declarations

Monday, March 29, 2010
1:37 PM

Kinds of Xschema declarations:

SimpleType: something that contains no other elements.

- xs:anyURI: A Uniform Resource Identifier
- xs:base64Binary: Base64-encoded binary data
- xs:boolean: true or false, 0 or 1
- xs:byte: integer >= -128 and <= 127
- xs:dateTime: date and time
- xs:duration: length of time in some units.
- xs:ID: a unique identifier for an element. From dynamic HTML ID's
- xs:IDREF: a reference to a unique ID. Cross-reference.
- xs:IDREFS: a list of references.
- xs:ENTITY: the name of an XML entity.
- xs:ENTITIES: a list of names of XML entities.
- xs:integer: positive or negative integer
- xs:decimal: signed number
- xs:language: same values as xml:lang
- xs:Name: an XML name
- xs:string: a text string.



Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

ComplexType: contains other elements, defined by nesting.

What's the big deal?

Monday, April 11, 2011
4:42 PM

If you declare an IDREF, then it must refer to a unique object.
(with a matching id).
If it doesn't, the document **fails to match the schema.**
Key idea: your program on the document **depends upon certain assumptions about the document**, and should not be run if the document doesn't comply.

Complex types

Monday, March 29, 2010
1:42 PM

Complex types:

A ComplexType is something that contains something other than just regular content.

Suppose you want to declare that my name can contain an attribute "language=en" or some such thing.

```
<xs:element name="fullName">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="language" type="xs:language"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

matches, e.g.,

```
<fullName language='en'>Alva Couch</fullName>
```

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Legend:

xs:complexType: a type that can contain attributes and other (sub-)elements.

xs:simpleContent: a declaration that sub-elements will not be included.

xs:extension: we will extend a known type via inheritance (xs:string).

Matching

An Xschema either "matches" or "does not match" a candidate XML file. In the preceding example:

```
<fullName language='en'>Alva</fullName>
```

matches, while

```
<fullName cat='dog'>Couch</fullName>
```

does not.

Matching is **syntactic**, not **semantic**.

Monday, March 29, 2010
1:53 PM

Complex types

- Any element type containing other elements (rather than text) is a "complex type".
- Suppose we want to define a document like this:

```
<address>
  <fullName>
    <first>Alva</first>
    <last>Couch</last>
  </fullName>
</address>
```

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

A complex schema:

Monday, March 29, 2010
1:54 PM

A schema for the preceding:

contents of xschema/document05.xsd...

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="address">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="fullName">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="first" type="nameComponent"/>
      <xs:element name="last" type="nameComponent"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name='language' type='xs:language'/>
 </xs:complexType>
</xs:element>

<xs:complexType name="nameComponent">
 <xs:simpleContent>
  <xs:extension base="xs:string"/>
 </xs:simpleContent>
</xs:complexType>

</xs:schema>
```

...end of xschema/document05.xsd

Pasted from
<http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?

include=style.txt>

Some new tricks

xs:sequence: allows us to specify a sequence of elements, must contain xs:element instances.

type="nameComponent": allows us to refer to another type declaration.

Monday, March 29, 2010
1:59 PM

Repeated and empty elements

contents of xschema/document06.xsd...

```
<?xml version='1.0'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="phonelist" type="myListType"/>
 <xs:complexType name="myListType">
  <xs:sequence>
   <xs:element name="phone" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="number" type="xs:string"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

...end of xschema/document06.xsd

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

MinOccurs='0': can be empty
MaxOccurs='unbounded': no length limit.

Matches, e.g.,

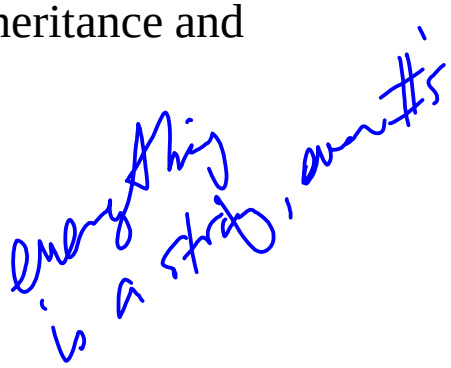
```
<phonelist>
 <phone number='555-1212'/>
 <phone number='1-800-food'/>
```

```
</phonelist>
```

Monday, March 29, 2010
2:01 PM

Facets:

- way of defining complex content by inheritance and restriction.
  - length, minLength, and maxLength
  - pattern (regular)
  - enumeration
  - whiteSpace
  - maxInclusive and maxExclusive
  - minInclusive and minExclusive
  - totalDigits
  - fractionDigits



Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

length facet

Monday, March 29, 2010
2:03 PM

length

- must often know how long one's input can be

```
<xs:simpleType name="shortString">
 <xs:restriction base="xs:string">
  <xs:maxLength value="10"/>
 </xs:restriction>
</xs:simpleType>
```

- xs:restriction: defines base type to restrict (xs:string).
- xs:maxLength: only 10 characters.

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:04 PM

whitespace facet

- determines how to handle whitespace in validating a document
  - collapse: replace multiple white space with one space
  - preserve: treat all whitespace literally
  - replace: every whitespace character becomes a space.
- This is done *before* validation!

```
<xs:simpleType name="shortString">
 <xs:restriction base="xs:string">
  <xs:maxLength value="10"/>
  <xs:whitespace value="collapse"/>
 </xs:restriction>
</xs:simpleType>
```

  matches any string that, after multiple spaces are replaced by one space, has no more than 10 characters in it!

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:05 PM

enumeration facet

- restricts a string to a set of values

```
<xs:simpleType name="locationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="work"/>
    <xs:enumeration value="home"/>
  </xs:restriction>
</xs:simpleType>
```

- This limits a location to be either "work" or "home"

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:06 PM

numeric facets

- control what kinds of numbers will be accepted in a document.
- Example:

```
<xs:simpleType name="onetoten">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>
```

  matches the numbers 1 to 10, inclusive.

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:07 PM

floating point facet

- format is ASCII!
- Example:

```
<xsd:simpleType name="myNumber">
  <xsd:restriction base="xs:decimal">
    <xsd:totalDigits value="5"/>
    <xsd:fractionDigits value="2"/>
  </xsd:restriction>
</xsd:simpleType>
```

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:13 PM

pattern facets

- Can also match general data patterns.

```
<xs:simpleType name="ssn">
  <xs:restriction base="xs:string">
    <xs:pattern value="\d\d\d-\d\d-\d\d\d\d"/>
  </xs:restriction>
</xs:simpleType>
```

  matches all social security numbers in the pattern 123-45-6789

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:08 PM

choice facet

- xs:choice allows one to select between alternatives for complex content:

```
<xs:element name="greeting">
  <xs:complexType mixed="true">
    <xs:choice>
      <xs:element name="hello"/>
      <xs:element name="hi"/>
      <xs:element name="dear"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

This matches

```
<greeting><hello/></greeting>
```

or

```
<greeting><hi/></greeting>
```

or

```
<greeting><dear/></greeting>
```

Monday, March 29, 2010
2:11 PM

all facet

- xs:all controls whether elements appear in *any* order:

```
<xs:element name="body">
 <xs:complexType mixed="true">
  <xs:all>
   <xs:element name="perpetrator"/>
   <xs:element name="victim"/>
   <xs:element name="cashAmount"/>
  </xs:all>
 </xs:complexType>
</xs:element>
```

  says that somewhere in mixed content, there should be three elements, perpetrator, victim, and cashAmount.

- E.g.

```
<body>
 The idiotic <perpetrator>Mr. Jones</perpetrator> stole
 <cashAmount>$1</cashAmount> from <victim>Mr. Stevens</victim>.
</body>
```

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

Monday, March 29, 2010
2:12 PM

any facet

- the xs:any label allows any reasonable content at all.

```
<xs:element name="notes" minOccurs="0">
 <xs:complexType>
  <xs:sequence>
   <xs:any namespace="http://www.w3.org/1999/xhtml"
       minOccurs="0" maxOccurs="unbounded"
       processContents="skip"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

  says that the notes element can contain arbitrary XHTML, and that we should not validate it.

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xschema.php?include=style.txt>

much, much more

Monday, March 29, 2010
2:14 PM

Much, much more:

- lists, sets, unions, etc.
- But the above representatives characterize the "power" of the specification language.
- In general, one can specify, fairly precisely, what an XML document or element should contain.
- Reason this is important: queries!
- XML represents state, you need to determine state.

Monday, March 29, 2010
2:15 PM

XPATH

A language for querying XML documents
Each query returns a list of elements.
Queries look like filesystem paths!

Monday, March 29, 2010
2:16 PM

Basic XPATH syntax

- *First use:* locating elements in some source document.
- *location:* where something is within the source document.
  - foo/bar matches an element named bar inside the content of an element named foo, without any intervening containing elements.
  - /foo/bar matches an element named bar inside the content of an element named foo, starting at top level of the document.
  - foo//bar matches an element named bar inside the content of an element named foo, at any level of nesting.
  - foo/@bar matches the attributes bar of tags named foo.
  - * matches any tag inside the current context.
  - ..: matches one level up (parent). foo/bar/.. matches any foo containing a bar.
  - .: matches current level: foo/bar/. is the same as foo/bar.

Pasted from
<http://www.cs.tufts.edu/comp/150WEB/notes/xpath.php?include=style.txt>

Examples of XPATH patterns

Monday, March 29, 2010
2:19 PM

- Examples: consider the document:
  contents of xpath/data.xml...

```
<?xml version="1.0"?>
<candies>
   <candy><name>taffy</name><price>$5</price>
      <length>1 ft</length></candy>
   <candy><name>chocolate</name><price>$3</price>
      <weight>1 lb</weight></candy>
</candies>
```

  ...end of xpath/data.xml
- What nodesets do the following match?
  - candies/candy/name
  - candies/candy/*
  - candy/*
  - candy/../*

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xpath.php?include=style.txt>

Monday, March 29, 2010
2:20 PM

Advanced XPATH syntax

- *conditions*: select locations that satisfy tests.
  - foo/bar[@title='cat'] matches a tag <bar> inside a tag <foo>, that happens to have an attribute 'title'whose value is 'cat'
  - bar[@title and @publisher] matches a tag <bar> that has both 'title' and 'publisher' attributes.
  - bar[what] matches a tag <bar> that *contains* a tag <what>.
- Examples:
  - candy[name='chocolate']
  - candy[name]
  - candy[2]
- *axes*: qualify how to interpret a name.
  - default is *child*: interpret names as children of current node.
  - self:: - current node.
  - parent:: - parent of current node.
  - preceeding-sibling:: - siblings before current node.
  - following-sibling:: - siblings after current node.
  - descendant-or-self:: - current node or descendants.
- Examples:
  - name[.='chocolate']/following-sibling::*
  - candy[name='chocolate']/following-sibling::candy

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xpath.php?include=style.txt>

Lots more functions available:

Monday, March 29, 2010
2:21 PM

Lots more functions available:

- Math: +,-,
  *,/,mod, sum(XPATH), floor(number), ceiling(number), round(number), etc.
- Type conversion: string(thing), number(thing)
- String: starts-with(string,prefix), contains(string,pattern), string-length(string), etc.
- Logic: not(bool), and, or.
- Comparison: =, !=, >, >=, <, <=. (yes, you must *escape* > and <).

Pasted from <http://www.cs.tufts.edu/comp/150WEB/notes/xpath.php?include=style.txt>

Monday, March 29, 2010
4:23 PM

How XSchemas and XPATHs interact

- An XSchema serves as a **validity filter** as to whether data is corrupt or reasonable.
  - If data does not match an XSchema, it is **discarded as corrupt.**
- XPATHs serve as **conditional expressions** to query the state of data.
  - An XPATH is considered "true" if it **matches a non-empty nodeset.**

So, what's the point?

- To describe business process,
  - We will use XML conforming to a specific XML Schema,
  - That contains XPATH statements that check data relationships,
  - And describes operations on data objects that have -- in turn -- their own schemas.
- Unfortunately, there are several different "standards" for doing this!

How we will use XSchemas and XPATHs

We will use XSchemas to ensure that data is valid before processing.

We will use XPATHs mainly as conditionals, e.g., "sale[@fruit]", which is "true" iff the referenced nodeset is **non-empty.**

We will write simple conditional statements in XPATH, e.g., "sum(@customers) > 3" to guide our "logic".

Symphonies, orchestration, and choreography

Monday, March 29, 2010
2:25 PM

Symphonies, orchestration, and choreography

Think of a business as a **symphony** to be performed.

To perform it, you need some form of **orchestration** as to who does what.

Alternatively, you can conceive of your business process as **choreography**: define interactions rather than sequences.

The use of orchestration versus choreography is a huge controversy;

Orchestration: big infrastructure (e.g., IBM Websphere) has global knowledge.

Choreography: little infrastructure (e.g., mashups) function from local knowledge.

I am on the choreography side in this controversy!

Suggested Readings:

- http://en.wikipedia.org/wiki/Business_process_modeling
- http://en.wikipedia.org/wiki/Business_Process_Execution_Language
- Primer on WS-BPEL: http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf
- http://en.wikipedia.org/wiki/Web_Service_Choreography
- Primer on WS-CDL: http://www.w3.org/TR/ws-cdl-10-primer/

And, for a really controversial view:

- http://www.infoq.com/minibooks/composite-software-construction