Database Systems
XML

H. Turgut Uyar Şule Öğüdücü

2005-2009

Topics

XML

- not a language itself
  - framework for defining languages
- XML-based languages
  - XHTML, DocBook, SVG, MathML, WML, XMI, ...
- XML-related languages
  - XPath, XQuery, XSL Transforms, SOAP, XLink, ...

XML Structure

- an XML document forms a *tree*
- nodes: elements, attributes, character data
  - root: *document element*
  - leaves: character data, self-closing elements

XML Example

Example (XHTML)

```
<html>
<head><title>ITU–SUNY ISE</title></head>
<body>
  <h1>Information Systems Engineering</h1>
  <p>You can get more information from the
    <a href="http://www.uolp.itu.edu.tr/">
      program page</a>.</p>
  <img src="logo.jpg" alt="ITU–SUNY ISE logo" />
</body>
</html>
```


XML Example

Example (DocBook)

```
<book lang="en">
  <title>Database Systems Project</title>
  <bookinfo>...</bookinfo>
  <chapter>...</chapter>
  <chapter>...</chapter>
  ...
</book>
```


XML Example

Example (DocBook)

```
<bookinfo>
  <author>
    <firstname>Mehmet</firstname>
    <surname>Ascii</surname>
  </author>
  <date>2007</date>
</bookinfo>
```


XML Example

Example (DocBook)

```
<chapter>
  <title>Introduction</title>
  <section>
    <title>Project Description</title>
    <para>This project ...</para>
  </section>
  ...
</chapter>
```

XML Example

Example (Movies)

```
<movies>
  <movie color="Color">
    <title>Usual Suspects</title>
    ...
  </movie>
  <movie color="Color">
    <title>Being John Malkovich</title>
    ...
  </movie>
  ...
</movies>
```


XML Example

Example (Movies)

```
<movie color="Color">
  <title>Usual Suspects</title>
  <year>1995</year>
  <score>8.7</score>
  <votes>35027</votes>
  <director>Bryan Singer</director>
  <cast>
    <actor>Gabriel Byrne</actor>
    <actor>Benicio Del Toro</actor>
  </cast>
</movie>
```


Advantages of XML

- text-based
  - independent from platform and application
- easy to process
  - ready-to-use parsers
- separate data from metadata
  - content and presentation

XML Compliance

- well-formed: forms a regular tree
- valid: conforms to a certain document structure
  - DTD, XML Schema

Document Type Definition Example

Example (Movie DTD)

```
<!DOCTYPE movies SYSTEM "movies1.dtd">
<movies>
  <movie color="Color">...</movie>
  <movie color="Color">...</movie>
  ...
</movies>
```


DTD Example

Example (Movie DTD)

```
<!ELEMENT movies (movie*)>
<!ELEMENT movie (title, year, score, votes,
                 director?, cast?)>
<!ATTLIST movie color (Color | BW) "Color">
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT score (#PCDATA)>
<!ELEMENT votes (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT cast (actor)+>
<!ELEMENT actor (#PCDATA)>
```


Parsing

- SAX: Simple API for XML
  - event-based
  - fast and less memory consuming
  - hard to program
- DOM: Document Object Model
  - keeps document in its entirety
  - easy to use in object oriented systems

Document Object Model

- traversing the tree:
  - getDocumentElement
  - getParentNode
  - getChildNodes
  - getNextSibling
  - getElementsByTagName
- getting attributes and values:
  - getAttribute
  - getTextContent
- modifying the tree:
  - appendChild removeChild
  - setAttribute

DOM Programming Example

Example (Movie class)

```
public class Movie {
    private String title;
    private boolean color;
    private int year;
    private float score;
    private int votes;
    private String director;
    private List cast;

    ...
}
```


DOM Programming Example

Example (Constructor from an XML element)

```
public Movie(Element me) {
  NodeList children = me.getChildNodes();
  this.title =
    children.item(0).getTextContent();
  this.year = Integer.parseInt(
    children.item(1).getTextContent());
  this.score = ...;
  this.votes = ...;
  this.director = ...;
  ...
}
```


DOM Programming Example

Example (Constructor from an XML element)

```
public Movie(Element me) {
  ...
  if (me.getAttribute(
        "color").compareTo("BW") == 0)
      this.color = false;
  else
      this.color = true;
  ...
}
```


DOM Programming Example

Example (Constructor from an XML element)

```
public Movie(Element me) {
  ...
  this.cast = new ArrayList();
  Element ce = (Element) me.getLastChild();
  NodeList nodes =
    ce.getElementsByTagName("actor");
  for (int i = 0; i < nodes.getLength(); i++) {
      Element ae = (Element) nodes.item(i);
      this.cast.add(ae.getTextContent());
  }
}
```

Validating Parser Example

Example (Parse from file and validate)

```
try {
    DocumentBuilderFactory xmlFactory =
      DocumentBuilderFactory.newInstance();
    xmlFactory.setValidating(true);
    DocumentBuilder xmlBuilder =
      xmlFactory.newDocumentBuilder();
    Document xmlDocument =
      xmlBuilder.parse("imdb1.xml");
} catch (SAXException e) {
    System.err.println("Invalid document.");
    e.printStackTrace();
}
```


DOM Programming Example

Example (Search movie by title)

```
/* get title of movie into movieTitle */
/* construct the documentElement */
NodeList nodes =
  documentElement.getElementsByTagName("title");
for (int i = 0; i < nodes.getLength(); i++) {
  Element te = (Element) nodes.item(i);
  String title = te.getTextContent();
  if (movieTitle.compareTo(title) == 0) {
    Element me = (Element) te.getParentNode();
    Movie m = new Movie(me);
  }
}
```


XPath Expressions

- path of nodes to find: chain of location steps
  - starting from the root (absolute)
  - starting from the current node (relative)
  - location steps are separated by / symbols

Example

- /movies/movie
- cast/actor or ./cast/actor
- ../../ year

Location Steps

- location step structure:
  axis :: node_selector [predicate]
- axis: where to search
- selector: what to search
- predicate: under which conditions

Axes

- child: all children, one level (default axis)
- descendant: all children, recursively (shorthand: //)
- parent: parent node, one level
- ancestor: parent nodes, up to document element
- attribute: attributes (shorthand: @)
- following-sibling: siblings that come later
- preceding-sibling: siblings that come earlier
- ...

Node Selectors

- node tag
- node attribute
- node text: text()
- all children: *

XPath Expression Examples

Example

- names of all directors:
  /child::movies/movie/director/text()
- all actors:
  movie/descendant::actor or ./movie//actor
- color info of all movies:
  //movie/attribute::color or //movie/@color
- scores of movies after this one:
  ./following-sibling::movie/score
- //actor/parent::movie/title
  //actor/../preceding-sibling::title

Predicates

- at a certain position: [index]
- existence of a child: [element]
- value of a child: [element="value"]
- existence of an attribute: [@attribute]
- value of an attribute: [@attribute="value"]

XPath Examples

Example

- the title of the first movie:
  /movies/movie[1]/ title
- all movies in the year 1997:
  movie[year="1997"]
- black-and-white movies:
  movie[@color="BW"]

XML Databases

- storing XML documents in columns (XMLDOC type)
  - make use of existing XML documents
  - operators for this type of columns
  - process documents in their entirety
  - if updates are not frequent, searches are simple
- distributing XML documents into different tables and columns
- native XML databases

Special DTD Attributes

- ID attributes
  - similar to primary keys
- IDREF attributes
  - similar to foreign keys

DTD Key Constraints

- key values always string
- keys always single-attribute
- key values unique in all of document
  - not just elements of the same type
- foreign keys refer only within the document

XML ID Attribute Example

Example (Movies)

```
<!DOCTYPE movies SYSTEM "movies2.dtd">
<movies>
  <movie color="Color">...  </movie>
  ...
  <person id="p302">Benicio Del Toro</person>
  <person id="p308">Gabriel Byrne</person>
  <person id="p639">Bryan Singer</person>
  ...
</movies>
```


XML ID Attribute Example

Example (Movies)

```
<movie color="Color">
  <title>Usual Suspects</title>
  ...
  <directorref id="p639" />
  <cast>
    <actorref id="p308" />
    <actorref id="p302" />
  </cast>
</movie>
```


DTD ID Attribute Example

Example

```
<!ELEMENT movies (movie*,person*)>
<!ELEMENT movie (title,...,directorref?,cast?)>
<!ATTLIST movie color (Color | BW) #IMPLIED>
<!ELEMENT title (#PCDATA)>
...
<!ELEMENT directorref EMPTY>
<!ATTLIST directorref id IDREF #REQUIRED>
<!ELEMENT cast (actorref)+>
<!ELEMENT actorref EMPTY>
<!ATTLIST actorref id IDREF #REQUIRED>
<!ELEMENT person (#PCDATA)>
<!ATTLIST person id ID #REQUIRED>
```


Reading Material

- Date (8th ed.)
  - Chapter 27: The World Wide Web and XML