Data Exchange

Contents:

- XML
  - Origins (HTML)
  - XML Schema
  - DOM, SAX
- Semantic Data Exchange
  - Integration Problems
  - MIX Model

Mariano Cilia – mcilia@gmail.com

Hyper-Text Markup Language

- HTML
- Hypertext:
  - A document that contains links to other documents
- Markup language:
  - A notation for writing text with markup tags
  - Tags indicate the structure of the text
  - Tags have names and attributes
  - Tags may enclose a part of the text

[Moeller, Schwartzwach]

HTML – Motivation

- Exchange data on the Internet
  - Documents are published by servers
  - Documents are presented by clients (browsers)
- HTML describes logical structure
  - Browsers are free to interpret markup tags
- HTML combined well-known ideas
  - Hyper-text (1945)
  - Markup languages (1970)

3

HTML Sample

```
<h1>Rhubarb Cobbler</h1>
<h2>Maggie.Herrick@bbs.mhv.net</h2>
<h3>Wed, 14 Jun 95</h3>
Rhubarb Cobbler made with bananas as the main sweetener.
It was delicious. Basicly it was
<table>
<tr><td> 2 1/2 cups <td> diced rhubarb
<tr><td> 2 tablespoons <td> sugar
<tr><td> 2 <td> fairly ripe bananas
<tr><td> 1/4 teaspoon <td> cinnamon
<tr><td> dash of <td> nutmeg
</table>
Combine all and use as cobbler, pie, or crisp.
Related recipes: <a href="#GardenQuiche">Garden
   Quiche</a>
```

4

Problems w/HTML

- Most HTML documents are invalid (with respect to the standard)
- The language is by design hard-wired to describe hypertext
  - Fixed collection of tags with fixed semantics
- Syntax and semantics is mixed together
  - The structure of data dictates its presentation in browsers
  - Different views are not supported

5

Cascading Style Sheets (CSS)

- Specify physical properties (layout) of tags
- Usually written in separate files
- Can be shared for many documents
- Advantages
  - data and layout are separated
  - document groups can have consistent looks
  - the look can easily be changed
- Check: http://www.csszengarden.com/ for powerful examples

6

SGML

- Standard Generalized Markup Language
- ISO standard, 1985
- huge amount of "document archive" applications in government, military, industry, academia, ...
- a successful well-known application: HTML is designed as a simple application of SGML

7

eXtensible Markup Language (XML)

XML is …

- Designed to separate syntax from semantics to provide a common framework for structuring information
  - Browser rendering semantics is completely defined by stylesheets
- Now **de facto standard**
  - W3C Recommendation 1998
- A simple subset of SGML, targeted for Web applications
  - Allow tailor-made markup for any imaginable application domain
- Platform independent

9

XML Sample

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
<author><email>Maggie.Herrick@bbs.mhv.net</email></author>
<date>Wed, 14 Jun 95</date>
<description>
Rhubarb Cobbler made with bananas as the main sweetener.
It was delicious.
</description>
<ingredients>
<item><amount>2 1/2 cups</amount><type>diced rhubarb</type></item>
<item><amount>2 tablespoons</amount><type>sugar</type></item>
<item><amount>2</amount><type>fairly ripe bananas</type></item>
<item><amount>1/4 teaspoon</amount><type>cinnamon</type></item>
<item><amount>dash of</amount><type>nutmeg</type></item>
</ingredients>
<preparation>
Combine all and use as cobbler, pie, or crisp.
</preparation>
<related url="#GardenQuiche">Garden Quiche</related>
</recipe>
```

10

Sample illustrate:

- the markup tags are chosen purely for **logical structure**
- this is just one choice of markup detail level
- we need to define which XML documents we regard as "recipe collections"
- we need a stylesheet to define browser presentation semantics
- we need to express queries in a general way

11

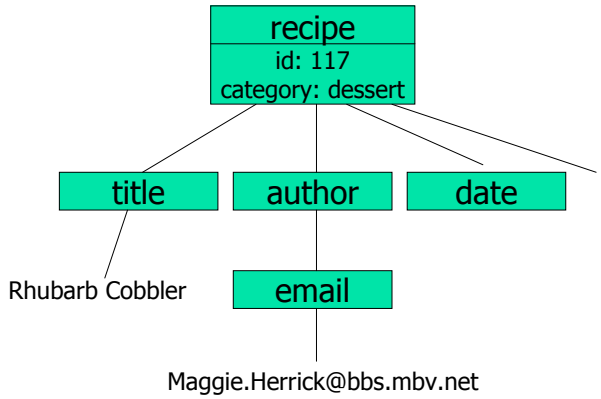
Conceptual View of XML

- An XML document is an **ordered, labeled tree**:
  - **character data** leaf nodes contain the actual data (text strings)
    - usually, character data nodes must be **non-empty** and **non-adjacent to other character data nodes**
  - **elements** nodes, are each labeled with
    - a name (often called the element *type*), and
    - a set of **attributes**, each consisting of a name and a value, and these nodes can have child nodes

12

A tree view…

- of the XML recipe collection



13

And Later

- **XML Schema** will later be used to define our class of recipe documents
- **XSLT** will be used to transform the XML document into XHTML (or HTML), including automatic construction of index, references, etc.
- **XLink**, **XPointer**, and **XPath** could be used to create cross-references
- **XQuery** will be used to express queries

14

XML Technologies

- the standard for representation of Web data
- by itself, just a notation for hierarchically structured text
- the real force of XML is generic languages and tools!
- by building on XML, you get a massive infrastructure for free
- See for a comprehensive list under:
  - **www.garshol.priv.no/download/xmltools**
  - **www.xmlsoftware.com**

15

To "use" XML

- Define your XML-based language
  - use e.g. XML Schema to define its syntax
- To build apps exploit
  - the generic XML tools (e.g. parsers, XSLT and XQuery processors),
  - the generic protocols, and
  - the generic programming frameworks (e.g. DOM or SAX)

16

XML Schema

What are XML Schemas?

- What's wrong with DTDs?
  - No data typing, especially for element content
  - Cannot enforce order and number of child elements
  - … among many others
- A document that describes what a correct document may contain
- Document syntax that describes the permissible content of XML docs
  - A **schema** is a definition of the syntax of an XML-based language
  - structured self-documentation

18

XML Schema supports

- **cardinality constraints** for sub-elements
- nil values (missing content)
- attribute and element **defaults**
- any-element, any-attribute
- **uniqueness** constraints and ID/IDREF attribute scope
- **regular expressions** for specifying valid chardata and attribute values
- lots of **built-in data types** for chardata and attribute values

19

Structure of XML Schema

- Data types
  - Simple types
  - Complex types
- Support for Namespaces
- Instances and schemas

20

Simple Types

- Cannot have children or attrs
- Built-in types: boolean, string, URIs, numeric, time
- Restricting: length, minLen, maxLen, totalDigits, …
- List and union types
- Type hierarchy: simple and complex can be derived (inherited) from other types

21

Complex types

- Can have child elements and attrs
  - Simple (character data)
    - <size system="EUROPEAN-DRESS">10</size>
  - Element (child element)

    <product manDate="2005-04-27">
    <number>4263</number>
    <size>10</size>
    </product>
  - Mixed
    - <letter>Dear <custName>John Doe</custName>… </letter>
  - Empty (no content)
    - <color value="blue"/>

22

Complex types (cont.)

- Content models
  - Order and structure of child elements
  - Sequence
    - Requires each child element to appear in the specified order
  - Choice
    - Requires exactly one of a group of specified elements to appear
  - All
    - Requires all the child elements to appear 0 or 1 times, in any order

23

Namespaces

- Namespaces are **declared** by special attributes and associated **prefixes**
  - **xmlns:***prefix***="***URI***"**
  - declares a namespace with a prefix and a URI

24

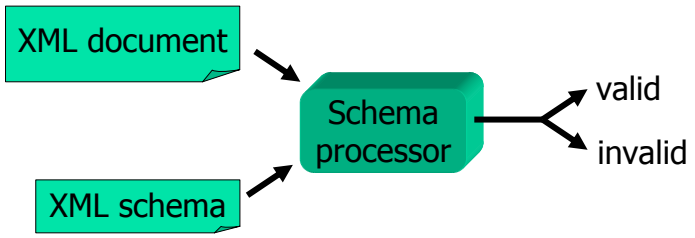
Schema Processing

- Given an XML document and a schema, a schema processor:
  - checks for validity, i.e. that the document conforms to the schema requirements
- The document being validated is called an instance document or application document

25

XML Correctness

- Well-formed
  - Obey XML syntactic rules
- Valid
  - Right type of elements and attributes, correct order and structure



26

Data Access: XPath

- a declarative language for **locating nodes and fragments** in XML trees
- used in:
  - XPointer (for **addressing**),
  - XSL (for **pattern matching**),
  - XML Schema (for **uniqueness and scope descriptions**), and
  - XQuery (for **selection and iteration**)

27

Data Access - XPath

```
<?xml version="1.0"?>
<A>
        <B x="1">aa</B>
        <C y="3">bb</C>
        <C y="4">cc</C>
        <D y="7">
                <E z="8">dd</E>
        </D>
</A>
```

e for accessing nodes in

mes separated by /

rchy of XML elements

/	
@	
[]	
*	

28

XQuery vs. XPath?

- Reminiscent, but different goals:
- XQuery:
  - SQL-like database queries
- XPath:
  - robust addressing into known information

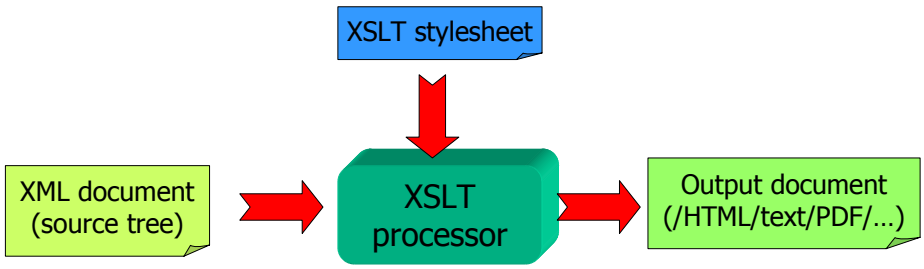
29

Doc Transformation: XSLT

- **XSL** (e**X**tensible **S**tylesheet **L**anguage)
- XSL Transformation
- an **XSLT stylesheet** is an XML document defining a **transformation** from one class of XML documents into another
  - Enhancing or reducing content
- XSLT is **not** intended as a completely general-purpose XML transformation language. Nevertheless it is generally useful

30

Doc Transformation: XSLT



31

Doc T XSLT

```
<xsl:stylesheet version="1.0" ...>
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection<
    <table border="1">
    <tr bgcolor="black">
      <th align="left">Title
      <th align="left">Artis
    </tr>
    <xsl:for-each select="c
    <tr>
      <td><xsl:value-of sel
      <td><xsl:value-of sel
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding.
<catalog>
  <cd>
    <title>Empire Burlesque</ti
    <artist>Bob Dylan</artist>
    <company>Columbia</company>
    <price>10.90</price>
  </cd>
.
.
</catalog>
```

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
Red	The Communards
Unchain my heart	Joe Cocker

XSLT processor → Output document (/HTML/text/PDF/...)

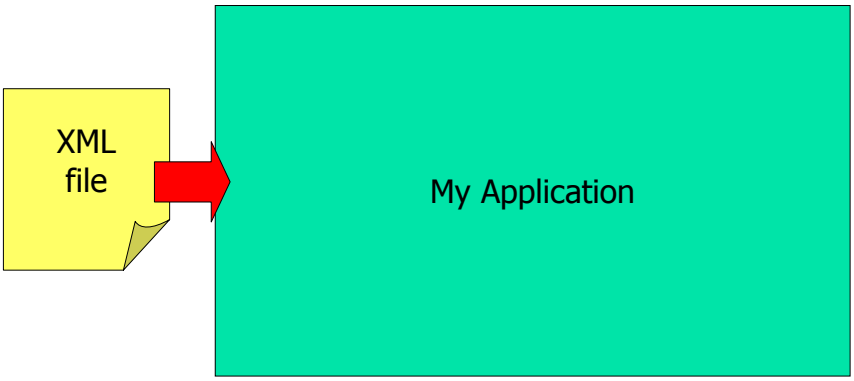
32

XML - Programming

- To work with XML in general-purpose programming languages we need to:
  - **parse** XML documents into XML trees
  - **navigate** through XML trees
  - **construct** (new) XML trees
  - **output** XML trees as XML documents
- DOM and SAX are corresponding APIs that are language independent and supported by numerous languages

33

DOM – Starting point



34

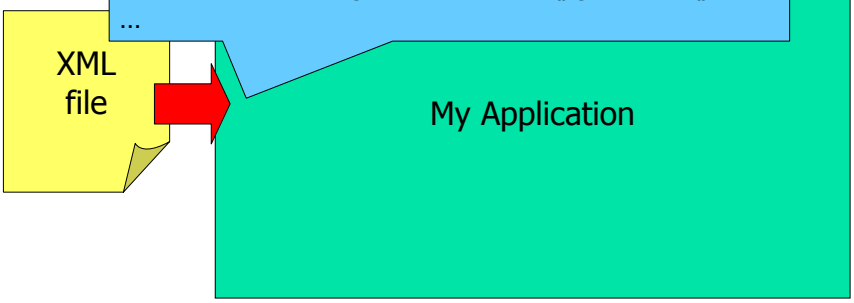
DOM

- It views an XML tree as a data structure
- The DOM API is specified in OMG IDL (Interface Definition Language)
- The whole XML document is represented (in main memory) using a tree
- DOM is too complicated to suit many programmers
  - Since it is a **general** API, it does not use **special** programming language features (for example, existing collection classes are ignored)
  - JDOM is designed to be **simple** and **Java-specific**

35

DOM Approach (1)

```
public static void main(String[] args) {
        DOMParser p = new DOMParser();
        p.parse( XMLFile );
        Document doc = p.getDocument();
        Node n = doc.getDocumentElement().getFirstChild();
...
```



36

DOM Approach (2)

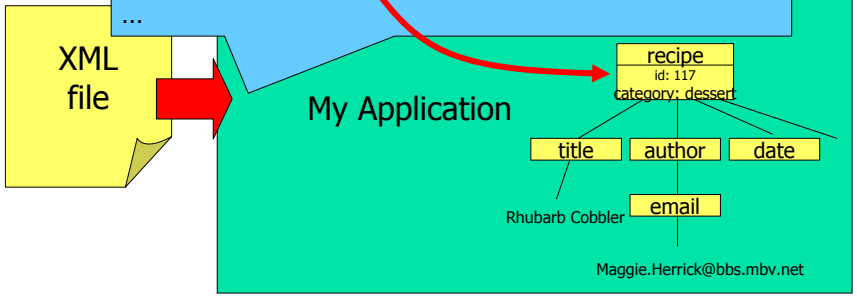
```
public static void main(String[] args) {
        DOMParser p = new DOMParser();
        p.parse( XMLFile );
```

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
<author><email>Maggie.Herrick@bbs.mhv.net</email></author>
<date>Wed, 14 Jun 95</date>
<description>
Rhubarb Cobbler made with bananas as the main sweetener.
It was delicious.
</description>
<ingredients>
<item><amount>2 1/2 cups</amount><type>diced rhubarb</type></item>
<item><amount>2 tablespoons</amount><type>sugar</type></item>
<item><amount>2</amount><type>fairly ripe bananas</type></item>
<item><amount>1/4 teaspoon</amount><type>cinnamon</type></item>
<item><amount>dash of</amount><type>nutmeg</type></item>
</ingredients>
<preparation>
Combine all and use as cobbler, pie, or crisp.
</preparation>
<related url="#GardenQuiche">Garden Quiche</related>
</recipe>
```

37

DOM Approach (3)

```
public static void main(String[] args) {
        DOMParser p = new DOMParser();
        p.parse( XMLFile );
        Document doc = p.getDocument();
        Node n = doc.getDocumentElement().getFirstChild();
...
```



38

SAX – Simple API for XML

- An XML tree is not viewed as a data structure, but as a stream of events generated by the parser
- Kinds of events are:
  - the start of the document is encountered
  - the end of the document is encountered
  - the start tag of an element is encountered
  - the end tag of an element is encountered
  - character data is encountered
  - a processing instruction is encountered
- Scanning the XML file from start to end, each event invokes a corresponding callback method that the programmer writes.

39

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String namespaceURI, String
localName, String qName, Attributes atts) {
        if (URI.equals("http://recipes.org") && …){
                …
…
```

XML file

My Application

40

SAX Approach (2)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
```

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
<author>
<email>
Maggie.Herrick@bbs.mhv.net
</email>
</author>
<date>Wed, 14 Jun 95</date>
<description>
Rhubarb Cobbler made with bananas
as the main sweetener.
It was delicious.
</description>
...
</recipe>
```

My Application

41

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
localName, String qName, Attribute
        if ( ... && ...){
                ...
...
```

XML file

My App

Trace of the startElement() invocations

42

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
                         Attribute
```

```
<recipe id="117" category="dessert">
```

```
My App
```

start document
starting element: recipe

43

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
                         Attribute
```

```
<recipe id="117" category="dessert">
<title>
```

```
My App
```

start document
starting element: recipe
-starting element: title

44

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
                         Attribute
```

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
```

```
My App
```

```
start document
starting element: recipe
-starting element: title
-end element: title
```

45

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
                         Attribute
```

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
<author>
<email>
Maggie.Herrick@bbs.mhv.net
</email>
</author>
<date>Wed, 14 Jun 95</date>
<description>
Rhubarb Cobbler made with bananas
as the main sweetener.
It was delicious.
</description>
...
</recipe>
```

```
My App
```

```
start document
starting element: recipe
-starting element: title
-end element: title
-starting element: author
--starting element: email
--end element: email
-end element: author
-starting element: date
...
end element: recipe
end document
```

46

SAX Approach (1)

```
public static void main(String[] args) {
        SAXParser p = new SAXParser();
        p.setContentHandler(f);
        p.parse( XMLFile );
}
public void startElement(String na
                         Attribute
```

```
<recipe id="117" category="dessert">
<title>Rhubarb Cobbler</title>
<author>
<email>
Maggie.Herrick@bbs.mhv.net
</email>
</author>
<date>Wed, 14 Jun 95</date>
<description>
Rhubarb Cobbler made with bananas
as the main sweetener.
It was delicious.
</description>
...
</recipe>
```

My App

```
start document
starting element: recipe
-starting element: title
-end element: title
-starting element: author
--starting element: email
--end element: email
-end element: author
-starting element: date
...
end element: recipe
end document
```

47

XML-related Technologies

- XML as standard basis for data exchange
- by building on XML, you get a massive infrastructure for free
- DOM and SAX (standard libraries) to manipulate XML documents from your programs
- Data Access: XPath
- Queries: XQuery
- Transformation: XSLT

48