eXtensible MarkUp Language

Ramu Parupalli

What Is XML?

- eXtensible Markup Language
- XML is Meta-markup language

  You define your own markup languages (tags) for your own problem domain
- A simplified version of SGML
- Maintains the most useful parts of SGML
- More flexible and adaptable than HTML
- Designed for interoperability with SGML as well as HTML
- Plays an important role in data exchange on the Web and in other application areas

Difference between XML and HTML

XML was designed to carry data, not displaying data

- XML is not a replacement for HTML.
- Different goals:

  XML was designed to describe data and to focus on what data is.

  HTML was designed to display data and to focus on how data looks.

- HTML is about displaying information, XML is about describing information.

Structure: HTML vs. XML

HTML (Automatic Presentation of Data)

```
<b> John Doe 1234 </b> // Display in bold
```

XML (Automatic Interpretation of Data)

```
<Employee>
<Name>
<firstName> John </firstName>
<lastName> Doe </lastName>
</Name>
<EmployeeID> 1234 </EmployeeID>
</Employee>
```

How Can XML be Used?

- XML is used in many aspects of web development, often to simplify data storage and sharing.
- XML is Used to Create New Internet Languages
- XML Makes Your Data More Available
- XML Simplifies Data Transport

XML Tree

- XML documents form a tree structure that starts at "the root" and branches to "the leaves".

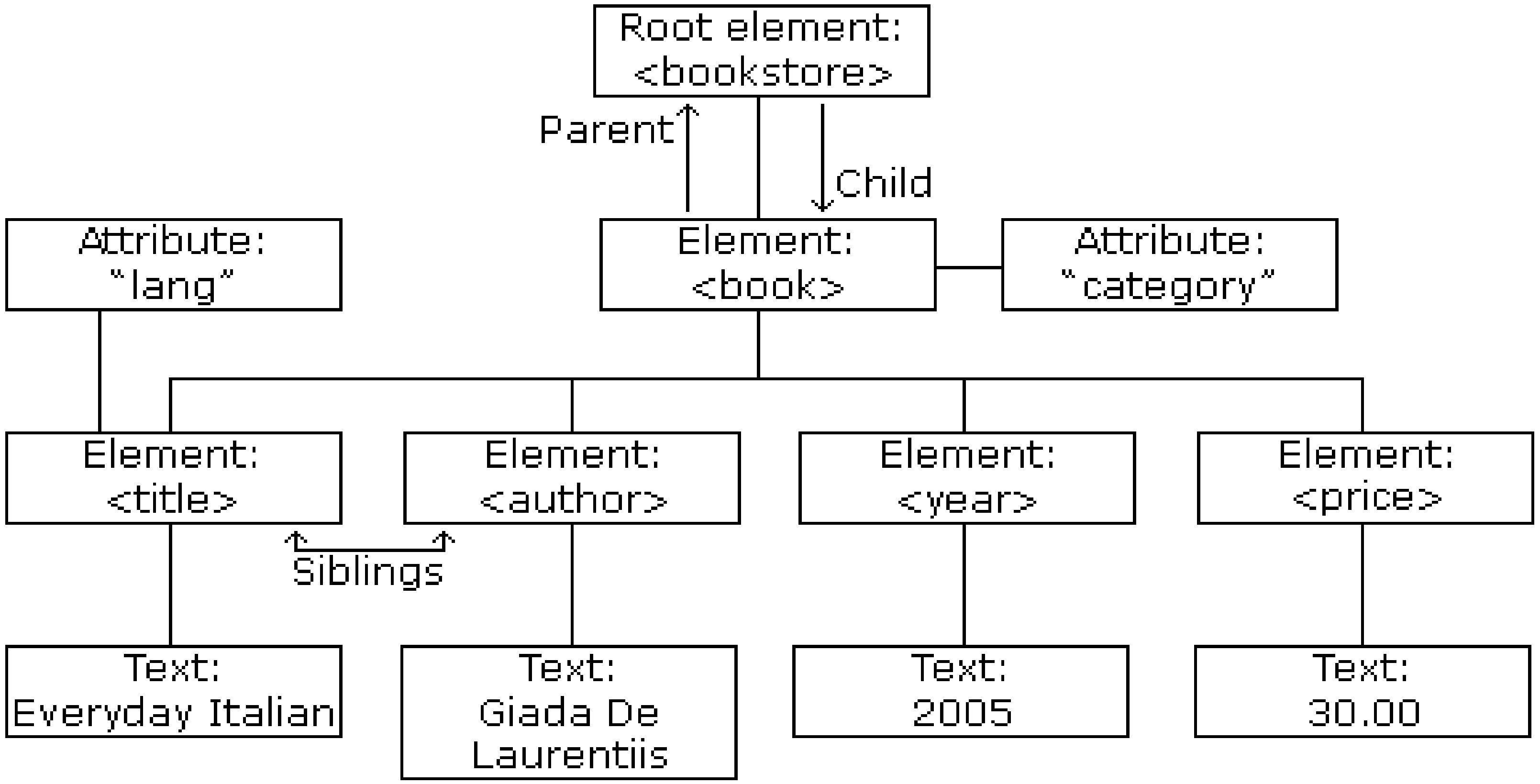
An example of XML

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```

Example2

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book  category="social“ >
        <title  lang="en“ >Everyday Italian</title>
         <author >Giada De laurentiis</author>
         <year >2005</year>
         <price >30</price>
    </book>
</bookstore>
```

Example


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De Laurentiis
Text:
2005
Text:
30.00

Why Is XML Important?

- Plain Text
  - Easy to edit
  - Useful for storing small amounts of data
  - Possible to efficiently store large amounts of XML data through an XML front end to a database
- Data Identification
  - Tell you what kind of data you have
  - Can be used in different ways by different applications

Why Is XML Important?

- Stylability
  - Inherently style-free
  - XSL---Extensible Stylesheet Language
  - Different XSL formats can then be used to display the same data in different ways
- Hierarchical
  - Faster to access
  - Easier to rearrange

XML Building blocks

- Element

  Delimited by angle brackets

  Identify the nature of the content they surround

  General format: <element> … </element>

  Empty element: <empty-Element/>

- Attribute

  Name-value pairs that occur inside start-tags after element name, like:
  <element attribute="value">

XML Building blocks--Prolog

- The part of an XML document that precedes the XML data
- Includes

  A declaration: version, [encoding, standalone]

  An optional DTD (Document Type Definition)
- Example

```
<?xml version="1.0" encoding=“UTF-8" standalone="yes"?>
```

XML Syntax

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML elements must be properly nested
- All XML documents must have a root tag
- Attribute values must always be quoted
- Comments in XML: <!-- This is a comment -->

XML Elements

- XML Elements are Extensible

  XML documents can be extended to carry more information

- XML Elements have Relationships

  Elements are related as parents and children

- Elements have Content

  Elements can have different content types: **element** content, **mixed** content, **simple** content, or **empty** content and **attributes**

- XML elements must follow the naming rules

XML Naming Rules

- XML elements must follow these naming rules:
- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces
- Any name can be used, no words are reserved except <xml>.

XML Element Content

- An XML element content is everything within and including the starting element to the closing element

Element Content Types

- Element content
- Mixed content
- Simple content
- Empty content
- An element attributes

Example

```
<rootelement>
<childelement1 attribute1=“An attribute value.”>An element value.</childelement1>
<childelement2>An element value.</childelement2>
<childelement3/>
<childelement4> An element value. <childchildelement1> An element value. </childchildelement1>
<childchildelement2> An element value. </childchildelement2>
</childelement4> </rootelement>
```

rootelement has element content, because it contains other elements

childelement4 has mixed content because it contains both text and other

childchildelement1 has simple content (or text content)

childelement3 has empty content, because it carries no information

Only the *childelement1* element has attributes

The attribute named *attribute1* has the value *An attribute value*

XML Attributes

- Located in the start tag of elements
- Provide additional information about elements
- Often provide information that is not a part of data
- Must be enclosed in quotes
- Should I use an element or an attribute?

metadata (data about data) should be stored as attributes, and that data itself should be stored as elements

Entity References

- Some characters have a special meaning in XML.
- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

Entity References

- <message>if salary < 1000 then</message>
- To avoid this error, replace the "<" character with an **entity reference**:
- <message>if salary &lt; 1000 then</message>

Entity References

- There are 5 predefined entity references in XML:

Entity References	**Character**
&lt;	<
&gt;	>
&amp;	&
&quot;	“
&apos;	‘

PCDATA

- PCDATA means parsed character data
- Text found between the start tag and the end tag of an XML element
- Text that will be parsed
- Tags inside the text will be treated as markup and entities will be expanded

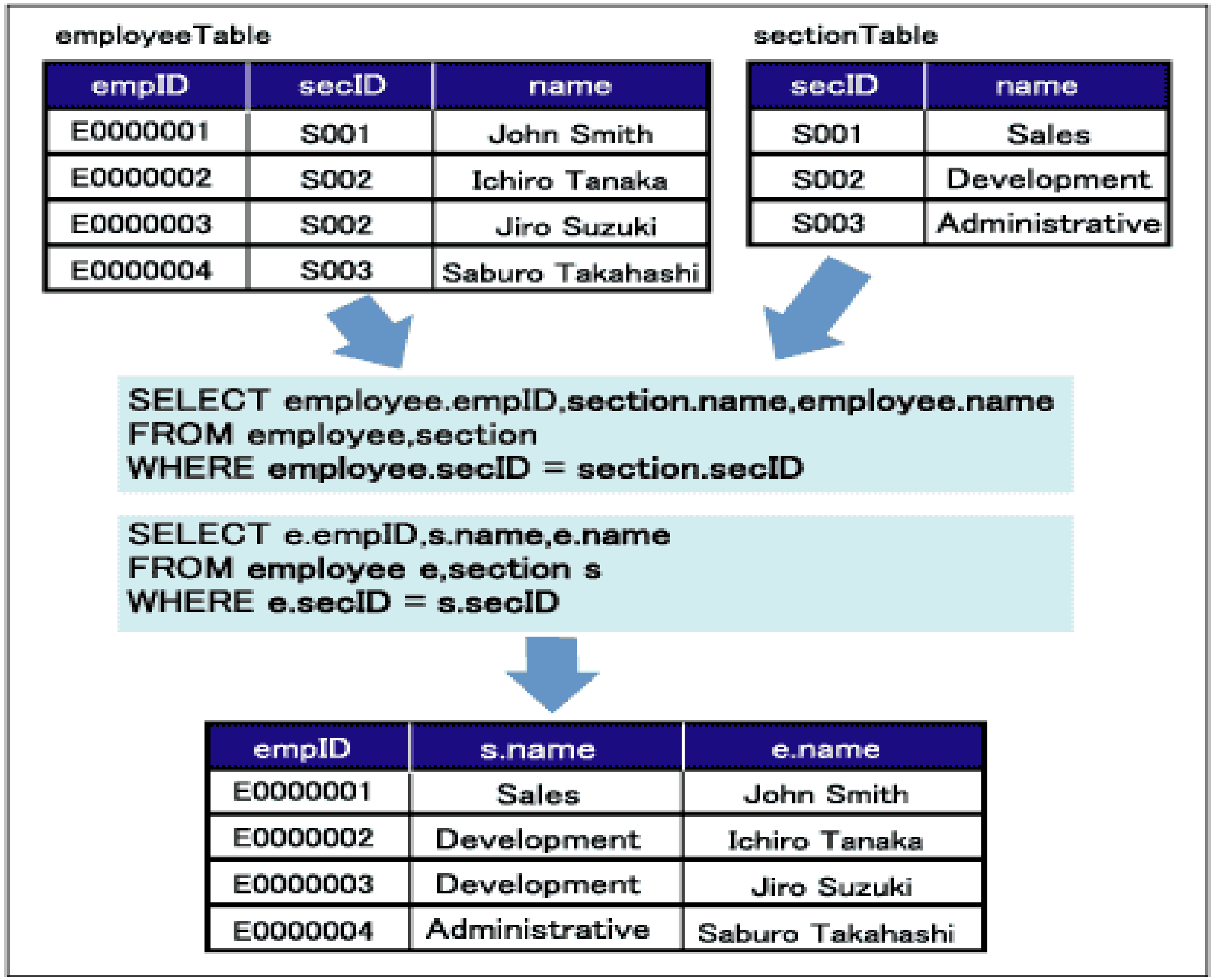
CDATA

- CDATA means character data
- Text that will NOT be parsed
- Tags inside the text will NOT be treated as markup and entities will not be expanded

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.

- ***What is Namespace? Why is it necessary?***
  - *Solving the Name Conflict Using a Prefix*
- ***Namespace Declaration***
- ***Namespace Declaration Scope***
- ***Namespaces with Attributes***
- ***Default Namespaces***
- ***Overwriting a Default Namespace***



employeeTable

empID	secID	name
E0000001	S001	John Smith
E0000002	S002	Ichiro Tanaka
E0000003	S002	Jiro Suzuki
E0000004	S003	Saburo Takahashi

sectionTable

secID	name
S001	Sales
S002	Development
S003	Administrative

```
SELECT employee.empID,section.name,employee.name
FROM employee,section
WHERE employee.secID = section.secID
```

```
SELECT e.empID,s.name,e.name
FROM employee e,section s
WHERE e.secID = s.secID
```

empID	s.name	e.name
E0000001	Sales	John Smith
E0000002	Development	Ichiro Tanaka
E0000003	Development	Jiro Suzuki
E0000004	Administrative	Saburo Takahashi

employeeXML Document

```
<employee>
<personInfo>
<empID>E0000001</empID>
<secID>S001</secID>
<name>John Smith</name>
</personInfo>
<personInfo>
 <empID>E0000002</empID>
<secID>S002</secID>
<name>Ichiro Tanaka</name>
</personInfo>
</employee>
```

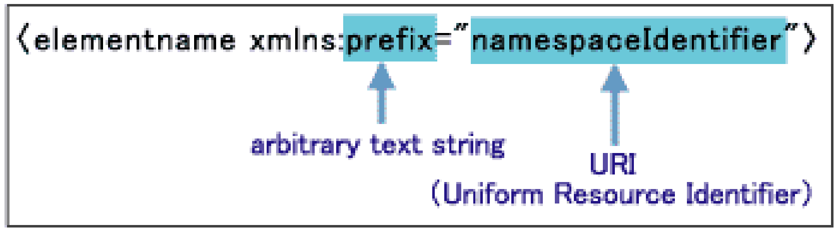
sectionXML Document

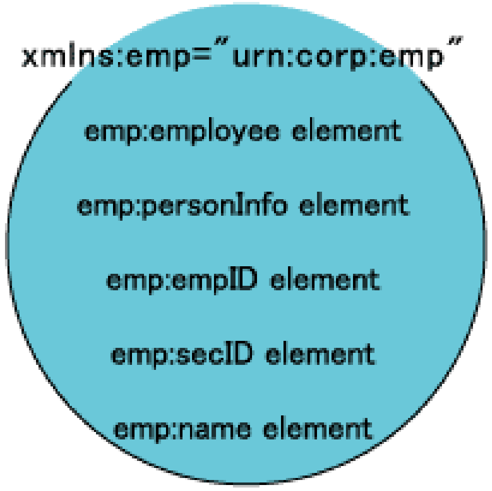
```
<section>
<sectionInfo>
 <secID>S001</secID>
<name>Sales</name>
 </sectionInfo>
 <sectionInfo>
 <secID>S002</secID>
 <name>Development</name>

 </sectionInfo>  </section>
```

employeeListXML Document

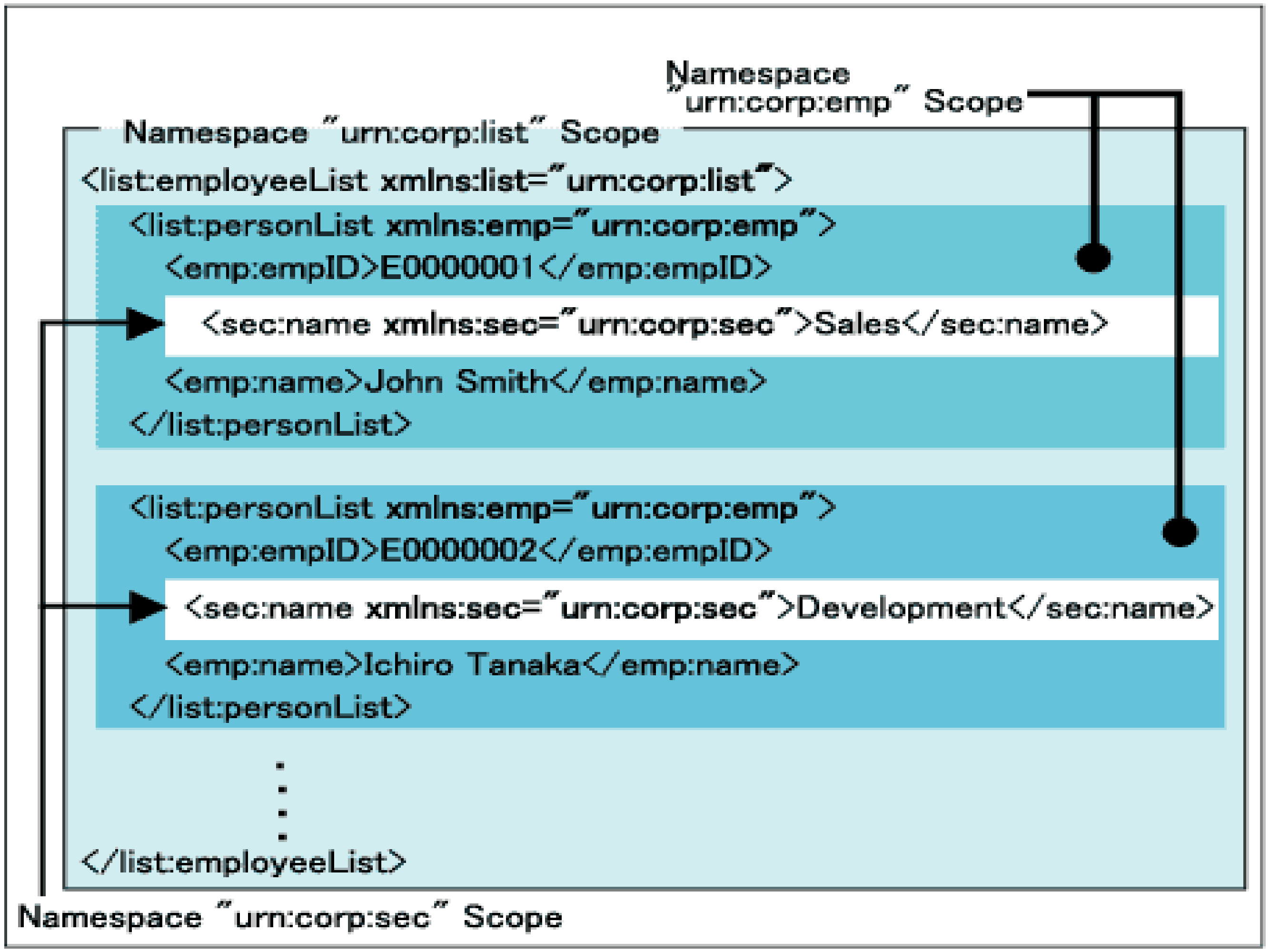
```
<employeeList>
<personList>
<empID>E0000001</empID>
<name>Sales</name>
<name>John Smith</name>
</personList> <personList>
<empID>E0000002</empID>
<name>Development</name>
<name>Ichiro Tanaka</name>
</personList>
</employeeList>
```


〈elementname xmlns:prefix="namespaceIdentifier"〉
arbitrary text string
URI
(Uniform Resource Identifier)



Elements belonging to the "urn:corp:emp" namespace

Namespace Declaration Scope


Namespace "urn:corp:emp" Scope
Namespace "urn:corp:list" Scope
Namespace "urn:corp:sec" Scope

```
<list:employeeList xmlns:list="urn:corp:list">
  <list:personList xmlns:emp="urn:corp:emp">
    <emp:empID>E0000001</emp:empID>
      <sec:name xmlns:sec="urn:corp:sec">Sales</sec:name>
    <emp:name>John Smith</emp:name>
  </list:personList>

  <list:personList xmlns:emp="urn:corp:emp">
    <emp:empID>E0000002</emp:empID>
     <sec:name xmlns:sec="urn:corp:sec">Development</sec:name>
    <emp:name>Ichiro Tanaka</emp:name>
  </list:personList>
              :
</list:employeeList>
```

```
<list:employeeList xmlns:list="urn:corp:list" xmlns:emp="urn:corp:emp "
 xmlns:sec="urn:corp:sec">
<list:personList>
 <emp:empID>E0000001</emp:empID>
 <sec:name>Sales</sec:name>
 <emp:name>John Smith</emp:name>
 </list:personList>
 <list:personList>
 <emp:empID>E0000002</emp:empID>
  <sec:name>Development</sec:name>
   <emp:name>Ichiro Tanaka</emp:name>
 </list:personList>
 <list:personList>
 <emp:empID>E0000003</emp:empID>
  <sec:name>Development</sec:name>
  <emp:name>Jiro Suzuki</emp:name>
  </list:personList>  <list:personList>
   <emp:empID>E0000004</emp:empID>
<sec:name>Administrative</sec:name>
<emp:name>Saburo Takahashi</emp:name>
  </list:personList>
</list:employeeList>
```

Namespaces with Attributes

Attribute belongs to same namespace as the element

```
<emp:employee xmlns:emp="urn:corp:emp">
    <emp:personInfo emp:empID="E0000001">
     <emp:secID>S001</emp:secID>
     <emp:name>John Smith</emp:name>
    </emp:personInfo>
   </emp:employee>
```

Attribute belongs to different namespace than the element

```
<emp:employee xmlns:emp="urn:corp:emp">
    <emp:personInfo work:empID="E0000001"
         xmlns:work="urn:corp:work">
     <emp:secID>S001</emp:secID>
     <emp:name>John Smith</emp:name>
    </emp:personInfo>
   </emp:employee>
```

Attribute does not belong to any namespace

```
<emp:employee xmlns:emp="urn:corp:emp">
    <emp:personInfo empID="E0000001">
     <emp:secID>S001</emp:secID>
     <emp:name>John Smith</emp:name>
    </emp:personInfo>
   </emp:employee>
```

- **The example below throws an error under the XML 1.0 specification**

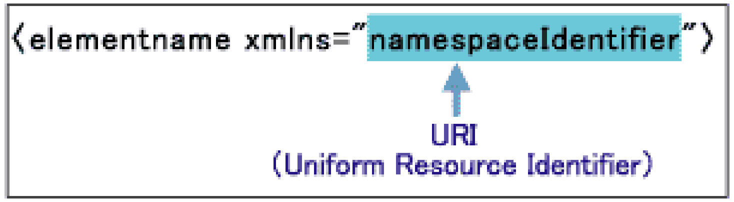
```
<employee>
    <personInfo empID="E0000001" empID="S0000001">
     <secID>S001</secID>
     <name>John Smith</name>
    </personInfo>
   </employee>
```

- The example below does not throw an error, since the Namespaces in XML specification is used

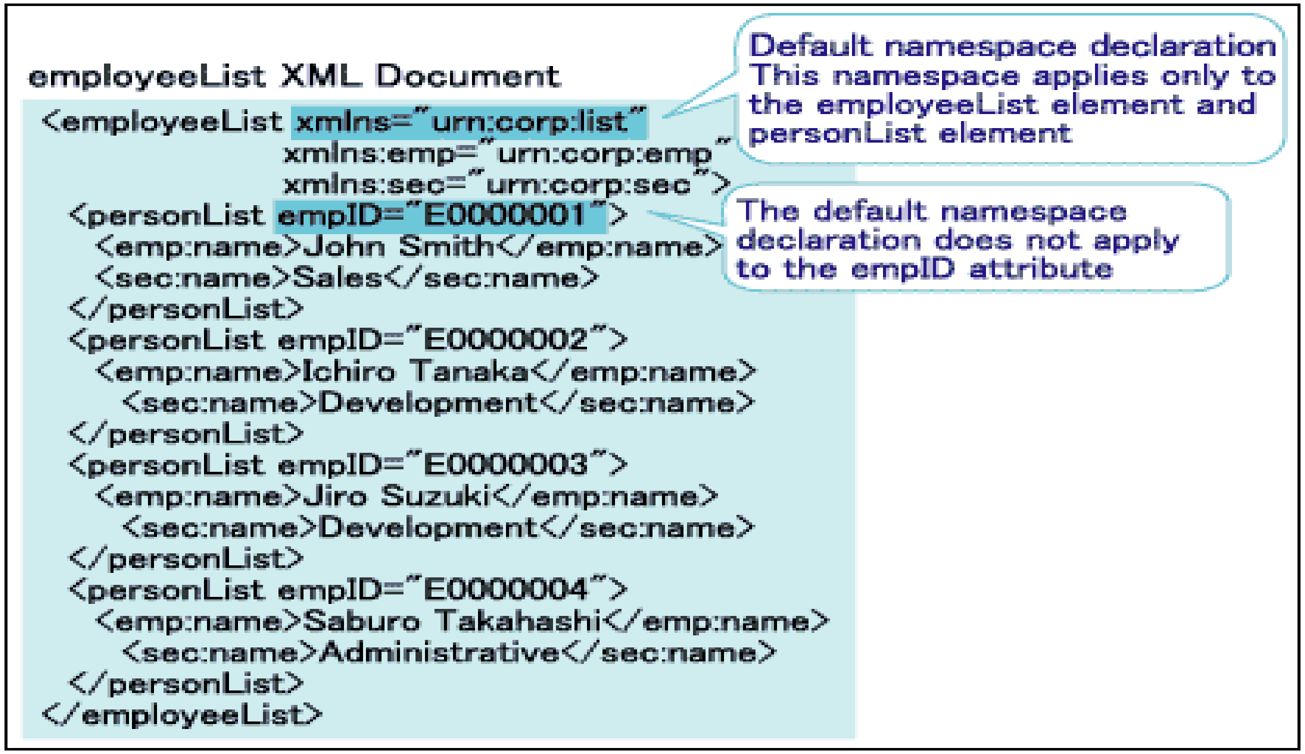
```
<emp:employee xmlns:emp="urn:corp:emp"
         xmlns:work="urn:corp:work">
    <emp:personInfo emp:empID="E0000001"
         work:empID="S0000001">
     <emp:secID>S001</emp:secID>
     <emp:name>John Smith</emp:name>
    </emp:personInfo>
   </emp:employee>
```

Default Namespaces

A "default namespace" is a namespace declaration that does not use a namespace prefix



```
<elementname xmlns="namespaceIdentifier">
```

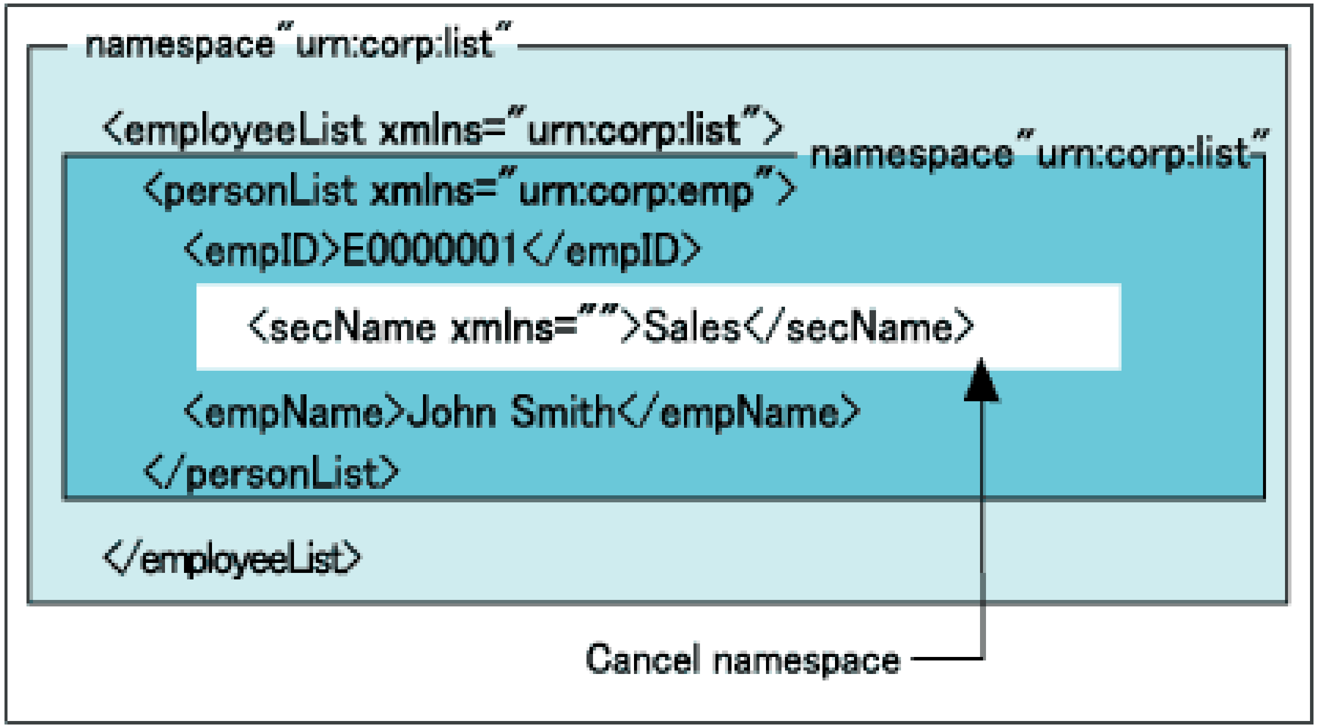


employeeList XML Document

```
<employeeList xmlns="urn:corp:list"
              xmlns:emp="urn:corp:emp"
              xmlns:sec="urn:corp:sec">
  <personList empID="E0000001">
    <emp:name>John Smith</emp:name>
    <sec:name>Sales</sec:name>
  </personList>
  <personList empID="E0000002">
    <emp:name>Ichiro Tanaka</emp:name>
      <sec:name>Development</sec:name>
  </personList>
  <personList empID="E0000003">
    <emp:name>Jiro Suzuki</emp:name>
      <sec:name>Development</sec:name>
  </personList>
  <personList empID="E0000004">
    <emp:name>Saburo Takahashi</emp:name>
      <sec:name>Administrative</sec:name>
  </personList>
</employeeList>
```

Default namespace declaration
This namespace applies only to the employeeList element and personList element
The default namespace declaration does not apply to the empID attribute

Overwriting a Default Namespace



```
<employeeList xmlns="urn:corp:list">
  <personList xmlns="urn:corp:emp">
    <empID>E0000001</empID>
      <secName xmlns="">Sales</secName>
    <empName>John Smith</empName>
  </personList>
</employeeList>
```

Well Formed Document

- An xml declaration should begin the document
- Include one or more elements
- Include Both Start and End Tags for Elements that aren't empty
- Close empty tags with />
- Root Elements must contains All other elements
- Nest Elements Correctly
- Use unique attribute names

XML Validation

- "Well Formed" XML document

  --correct XML syntax
- "Valid" XML document
  - "well formed"
  - Conforms to the rules of a DTD (Document Type Definition)
- XML DTD
  - defines the legal building blocks of an XML document
  - Can be inline in XML or as an external reference
- XML Schema
  - an XML based alternative to DTD, more powerful
  - Support namespace and data types

Displaying XML

- XML documents do not carry information about how to display the data

- We can add display information to XML with
  - CSS (Cascading Style Sheets)
  - XSL (eXtensible Stylesheet Language) --- preferred

XML Application1—Separate data

XML can Separate Data from HTML

- Store data in separate XML files
- Using HTML for layout and display
- Using Data Islands
- Data Islands can be bound to HTML elements

Benefits:

Changes in the underlying data will not require any changes to your HTML

XML Application2—Exchange data

XML is used to Exchange Data

- Text format
- Software-independent, hardware-independent
- Exchange data between incompatible systems, given that they agree on the same tag definition.
- Can be read by many different types of applications

Benefits:

- Reduce the complexity of interpreting data
- Easier to expand and upgrade a system

XML Application3—Store Data

XML can be used to Store Data

- Plain text file
- Store data in files or databases
- Application can be written to store and retrieve information from the store
- Other clients and applications can access your XML files as data sources

Benefits:

Accessible to more applications

XML Application4—Create new language

XML can be used to Create new Languages

- WML (Wireless Markup Language) used to markup Internet applications for handheld devices like mobile phones (WAP)
- MusicXML used to publishing musical scores
- Many More.

XML support in IE 5.0+

Internet Explorer 5.0 has the following XML support:

- Viewing of XML documents
- Full support for W3C DTD standards
- XML embedded in HTML as Data Islands
- Binding XML data to HTML elements
- Transforming and displaying XML with XSL
- Displaying XML with CSS
- Access to the XML DOM (Document Object Model)

*Netscape 6.0 also have full XML support

Microsoft XML Parser

- Comes with IE 5.0+
- The parser features a language-neutral programming model that supports:
  - JavaScript, VBScript, Perl, VB, and more
  - W3C XML 1.0 and XML DOM
  - DTD and validation

XML Standards

- XML, DTD
- XSL, XSLT, XPath
- DOM, SAX
- W3C XML Schema
- Namespaces
- XLink, XPointer
- XHTML
- XQL

Domain-specific XML Standards

- Chemical - CML
- 2D Graphics - SVG
- Math - MathML
- Music - MusicML
- Travel -OTA
- Many more ...

http://xml.org/xmlorg_registry/index.shtml

How XML Is Used in the Real World

Bank Internet Payment System (*BIPS*)

Banking Industry Technology Secretariat (*BITS*)

Financial Exchange (*IFX*)

Geography Markup Language (*GML*)

Human Resources Background Checks and Payroll Deductions Language (HR-*XML*)

Windows Rights Management Services (*RMS*) by Microsoft

XML Process Definition Language (*XPDL*) for workflow management

Schools Interoperability Framework (*SIF*)

Using *XML*: Mathematical Markup Language

Mathematical Markup Language, MathML, was designed to let people embed mathematical and scientific equations in Web pages

This document just displays the equation $4x^2 - 5x + 6 = 0$

```
<?xml version="1.0"?>
 <math xmlns="http://www.w3.org/1998/Math/MathML">
 <mrow>
<mrow>
 <mn>4</mn> <mo>&InvisibleTimes;</mo>
<msup><mi>x</mi><mn>2</mn>
</msup>
<mo>-</mo>
 <mrow> <mn>5</mn>
 <mo>&InvisibleTimes;</mo>
 <mi>x</mi></mrow>
<mo>+</mo> <mn>6</mn>
</mrow> <mo>=</mo>
 <mn>0</mn>
</mrow>
</math>
```

Using *XML*: Chemical Markup Language:-

Chemical Markup Language (*CML*) was developed by Peter Murray-Rust and lets you view three-dimensional representations of molecules in a Jumbo browser. Using *CML*, one chemist can publish a visual model of a molecule and exchange that model with others.

```
<molecule xmlns="http://www.xml-cml.org" id="formamide">
<atomArray> <stringArray builtin="atomId">H1 C1 O1 N1 Me1 Me2</stringArray>
<stringArray builtin="elementType">H C O N C C</stringArray>
<integerArray builtin="hydrogenCount">0 1 0 1 3 3</integerArray>
</atomArray>
<bondArray> <stringArray builtin="atomRef">C1 C1 C1 N1 N1</stringArray>
<stringArray builtin="atomRef">H1 O1 N1 Me1 Me2</stringArray>
<stringArray builtin="order">1 2 1 1 1</stringArray>
 </bondArray>
 <h:html xmlns:h="http://www.w3.org/TR/html20">
 <p>Formamide is the simplest amide ...</p>
 <p> This represents a <emph>connection table</emph> for formamide. The structure corresponds to the
        diagram: </p>
       <pre>H3 H1 \ / N1-C1=O1 / H2</pre>
       </h:html> <float title="molecularWeight" units="g">45.03</float> <list title="local information">
       <!-- <link title="safety" href="/safety/chemicals.xml#formamide">
       </link> --> <string title="location">Storeroom 12.3</string>
 </list>
</molecule>
```

Using *XML*: Synchronized Multimedia Integration Language:

Synchronized Multimedia Integration Language (*SMIL*, pronounced "smile") lets you customize multimedia presentations. We'll even be able to create *SMIL* files that can be run in RealNetwork's RealPlayer (now called RealOne).

For example, here's the beginning of a *SMIL* document that plays background music and displays a slide show of images and text:

```
<?xml version="1.0"?>
 <!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN" "http://www.w3.org/TR/REC-
smil/SMIL10.dtd">
<smil>
<body>
 <par id="show">
 <audio src="river.wav" region="background_audio" type="audio/x-wav" dur="20s"/>
 <seq id="slides">
<par id="slide01">
<img src="mountain.jpg" type="image/jpeg" dur="5s"/>
<text src="welcome.txt" type="text/plain" dur="5s"/>
</par>
```

Using *XML*: Scalable Vector Graphics:

A number of popular *XML* applications revolve around graphics, and one of these applications is Scalable Vector Graphics (*SVG*), a W3C-based *XML* application.

You can see a sample *SVG* document, which draws a blue ellipse filled in with light blue color.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
 <title>SVG Example</title>
 <ellipse cx="200" cy="100" rx="100" ry="60" style="fill:lightblue; stroke:blue;
      stroke-width:6"/>
</svg>
```

Core Java APIs for XML

- JAXP: Parsing and Transforming
- JAXB: High-level XML programming
- JAXM: Messaging
- JAXR: Registry APIs
- JDOM: Java-optimized Parsing

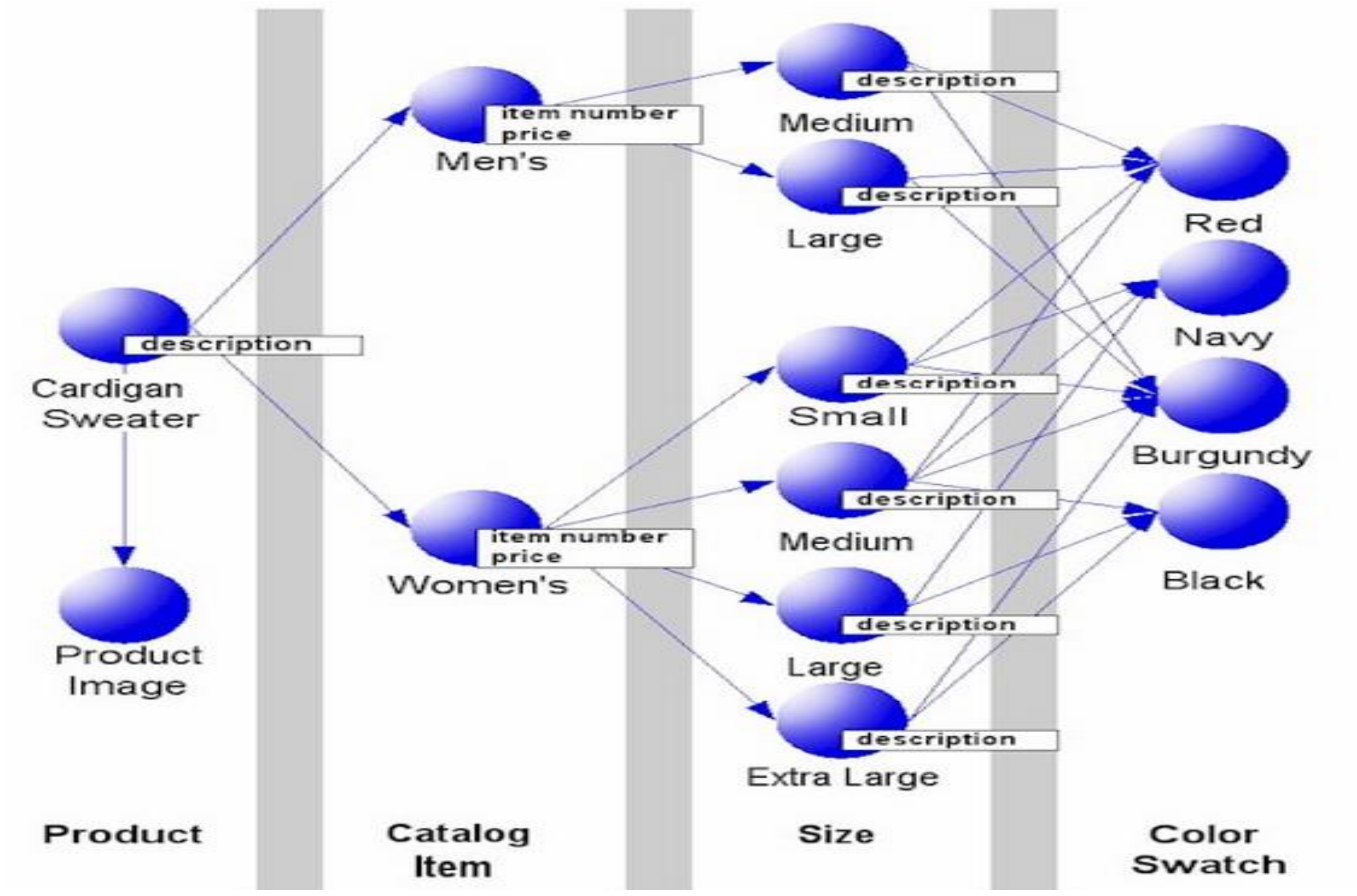
E-Commerce Standards

- ebXML
- UDDI (Universal Description,Discovery and Integration)
- SOAP (Simple Object Access Protocol)
- W3C XP (XML Protocol)
- WSDL (Web Services Description Lang.)
- S2ML (Security Services ML)
- XAML (Transaction Authority ML)

Conclusion

- XML is a self-descriptive language
- XML is a powerful language to describe structure data for web application
- XML is currently applied in many fields
- Many vendors already supports or will support XML

Complex Data



Thank you

Any Questions?

ramup@cdac.in