Introduction to XML

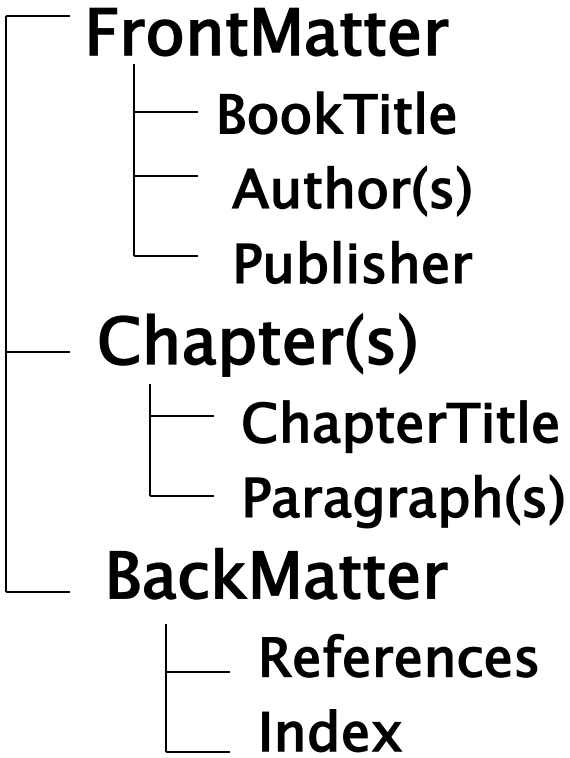
Chunyan Ji
United International College
2016 Fall

What is XML

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined.
- You must define your own tags
- XML is designed to be self-descriptive

XML to describe a book

A Book



XML fragment of the Book

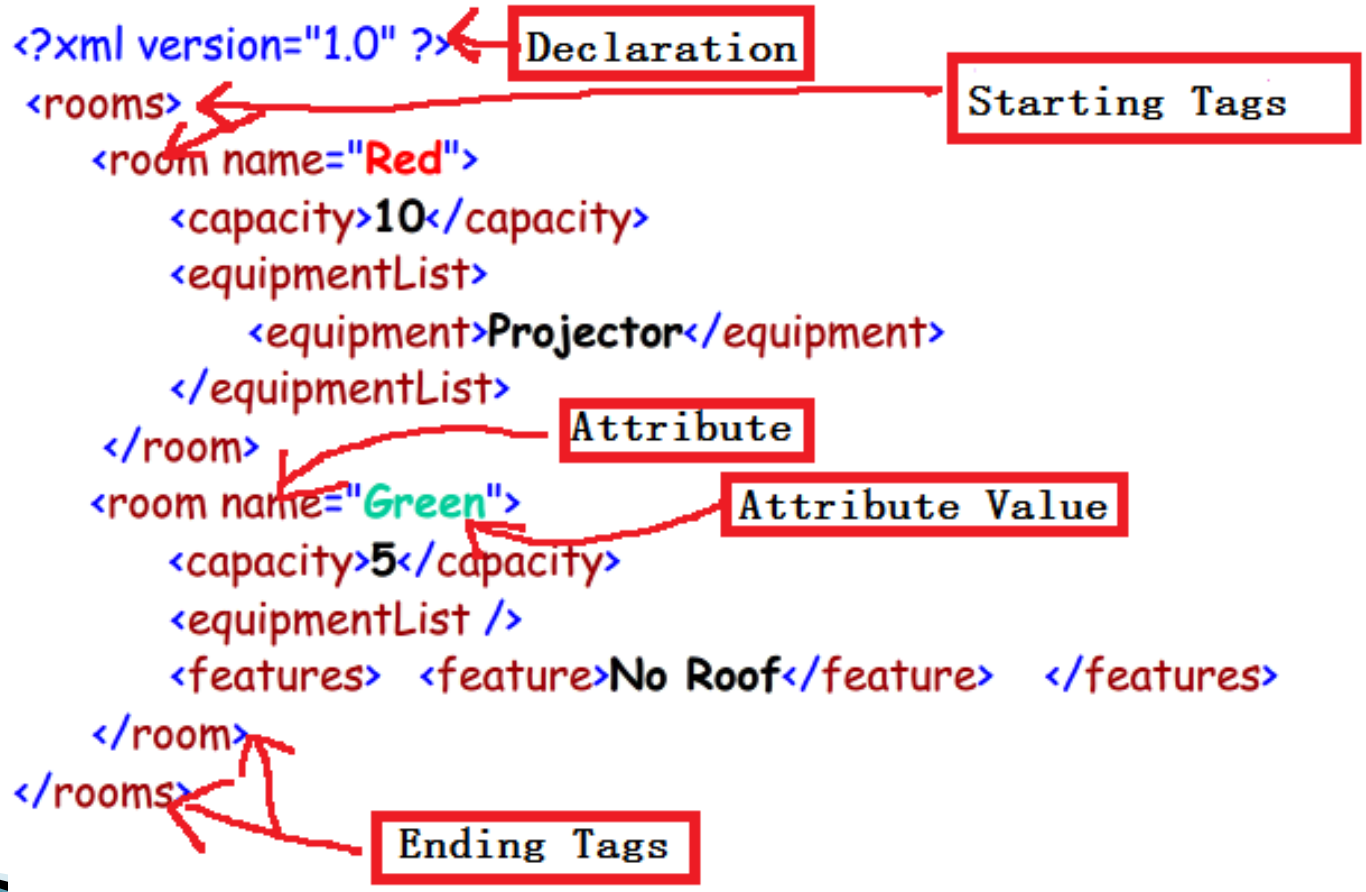
```
<Book>
  <FrontMatter>
    <BookTitle>Essential Java</BookTitle>
    <Author>Adam Smith</Author>
    <Author>Tom White</Author>
    <Publisher>Educattion Press</Publisher>
  </FrontMatter>
  <Chapter>
    <ChapterTitle>What is OOP</ChapterTitle>
    <Paragraph>What is Object</Paragraph>
  </Chapter>
 <Chapter>
      …
  </Chapter>
</Book>
```

What does xml look like?

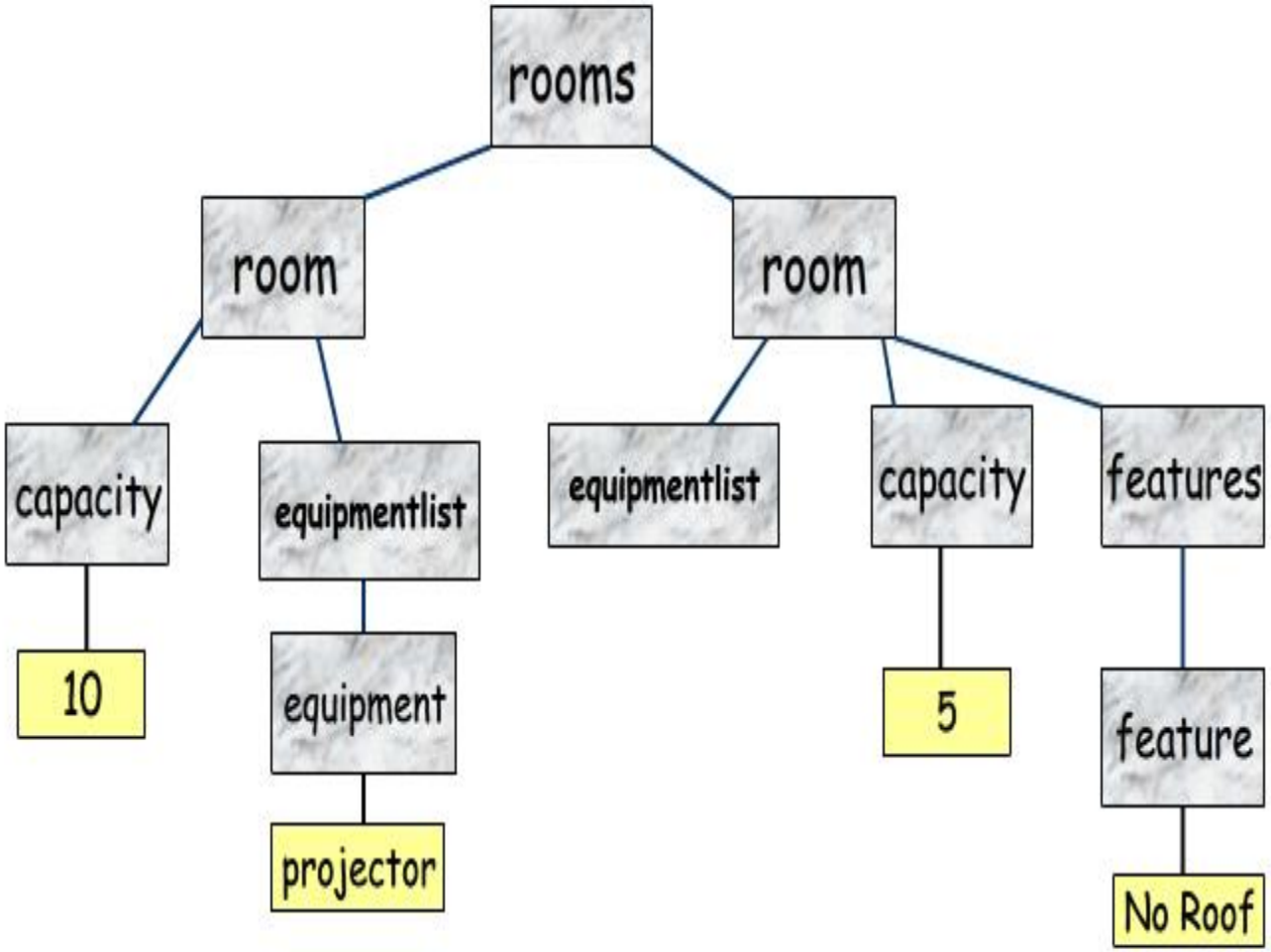
```
<?xml version="1.0" ?>
 <rooms>
     <room name="Red">
         <capacity>10</capacity>
         <equipmentList>
             <equipment>Projector</equipment>
         </equipmentList>
      </room>
     <room name="Green">
         <capacity>5</capacity>
         <equipmentList />
         <features>  <feature>No Roof</feature>   </features>
     </room>
</rooms>
```

mark.baker@computer.org

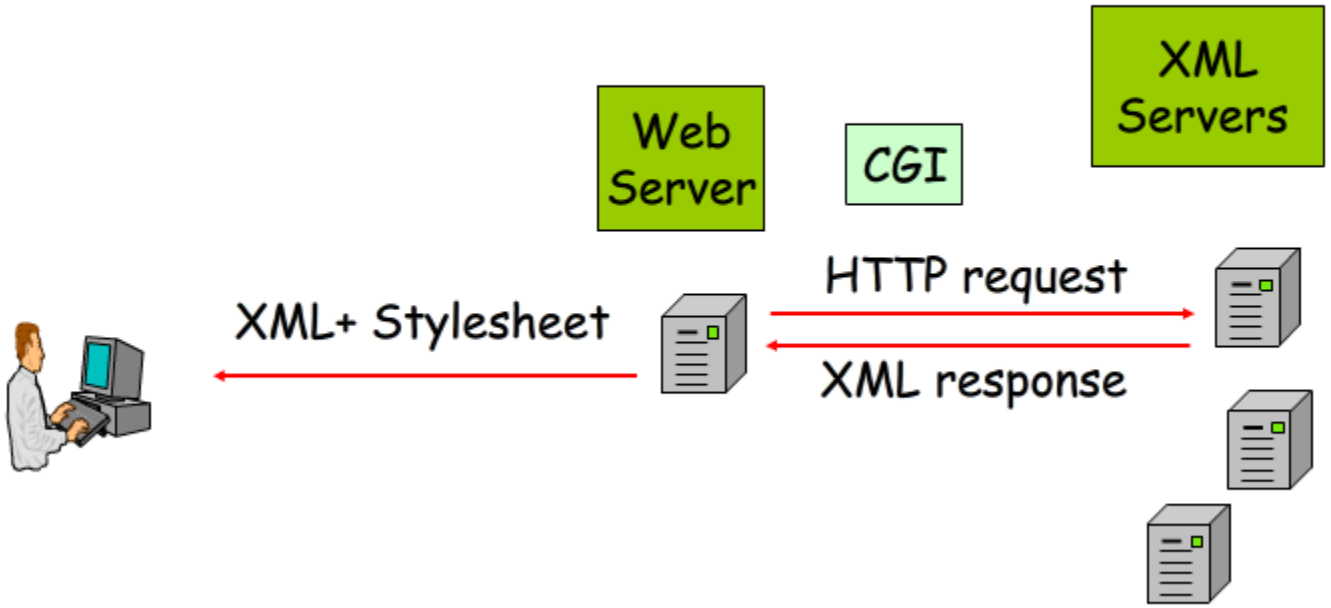
Declaration – Tags – Attributes



XML and Tree


rooms
room
room
capacity
equipmentlist
equipmentlist
capacity
features
10
equipment
5
feature
projector
No Roof

How can XML be used?



XML Syntax

- XML declaration: <?xml version="1.0“ encoding="UTF-8"?>
- Must have a root element
- All XML elements must have a closing tag
- Case Sensitive
- Attributes must be quoted
- <!-- comments -->

Is this XML?

```
<Poem>
  <Moon>
    <Line><Author>李白</Line>
    <Line>床前明月光</Line>
    <Line>疑是地上霜</Line>
    <Line>举头望明月</Line>
    <Line>低头思故乡</Author></Line>
  </Moon>
</Poem>
```

XML, HTML, & XHTML

- HTML—display oriented
- XML
  - Element set is fully extensible
  - Syntax is fixed
- XHTML—HTML modified to be XML compliant

Elements

- Elements are markup that enclose content
  - <element_name>…</element_name>

Attributes

- Associate a name-value pair with an element

- <tag name1="value1"
  name2='value2'>…</tag>

```
<book color="red">Java Book</book>
<book color="blue">C book</book>
```

Comments

- **Same as HTML comment**

```
<!-- This is a comment -->
```

XML Programming in C# and .NET

- What we are going to focus on is XML programming
- Create XML using C#
- Writing it to file
- Load the XML file to memory
- Use XPath to SelectNodes, SelectSingleNode
- Get/Set value to the node/attribute.

XPath Programming

- To get the value of a node, you need to select the node first
- Using xmlDoc.SelectSingleNode function or xmlDoc.SelectNodes function
- The string used in these functions is the Xpath: *xmlDoc.SelectNodes("//Product_id");*
- The basic XPath syntax is similar to filesystem addressing
- If the path starts with the slash / , then it represents an absolute path to the required element.

/AAA

Select the root element AAA

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC/>
</AAA>
```

/AAA/CCC

Select all elements CCC which are children of the root element AAA

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC/>
</AAA>
```

/AAA/DDD/BBB

Select all elements BBB which are children of DDD which are children of the root element AAA

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC/>
</AAA>
```

If the path starts with // then all elements in the document which fulfill following criteria are selected.

//BBB
Select all elements BBB

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
      </DDD>
   </CCC>
</AAA>
```

//DDD/BBB

Select all elements BBB which are children of DDD

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
      </DDD>
   </CCC>
</AAA>
```

The star * selects all elements located by path

/AAA/CCC/DDD/*
Select all elements enclosed by elements /AAA/CCC/DDD

```
<AAA>
   <XXX>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </XXX>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </CCC>
   <CCC>
      <BBB>
         <BBB>
            <BBB/>
         </BBB>
      </BBB>
   </CCC>
</AAA>
```

/AAA/BBB[last()]

Select the last BBB child of element AAA

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
</AAA>
```

- Attributes are specified by @ prefix.

//@id
Select all attributes @id
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

//BBB[@id]

Select BBB elements which have attribute id

```
<AAA>
    <BBB id = "b1"/>
    <BBB id = "b2"/>
    <BBB name = "bbb"/>
    <BBB/>
</AAA>
```

//BBB[@name]

Select BBB elements which have attribute name

```
<AAA>
    <BBB id = "b1"/>
    <BBB id = "b2"/>
    <BBB name = "bbb"/>
    <BBB/>
</AAA>
```

//BBB[@*]

Select BBB elements which have any attribute

```
<AAA>
    <BBB id = "b1"/>
    <BBB id = "b2"/>
    <BBB name = "bbb"/>
    <BBB/>
</AAA>
```

//BBB[not(@*)]

Select BBB elements without an attribute

```
<AAA>
    <BBB id = "b1"/>
    <BBB id = "b2"/>
    <BBB name = "bbb"/>
    <BBB/>
</AAA>
```

```
//BBB[@id='b1']
```

Select BBB elements which have attribute id with value b1

```
<AAA>
   <BBB id = "b1"/>
   <BBB name = " bbb "/>
   <RRR name = "hhh"/>
</AAA>
```

```
//BBB[@name='bbb']
```

Select BBB elements which have attribute name with value 'bbb'

```
<AAA>
    <BBB id = "b1"/>
    <BBB name = " bbb "/>
    <BBB name = "bbb"/>
</AAA>
```

```
//BBB[normalize-space(@name)='bbb']
```

Select BBB elements which have attribute name with value bbb, leading and trailing spaces are removed before comparison

```
<AAA>
    <BBB id = "b1"/>
    <BBB name = " bbb "/>
    <BBB name = "bbb"/>
</AAA>
```

Practice

- Refer to CS_XML.doc