Application Design Using Java

Lecture 15

Property Maps

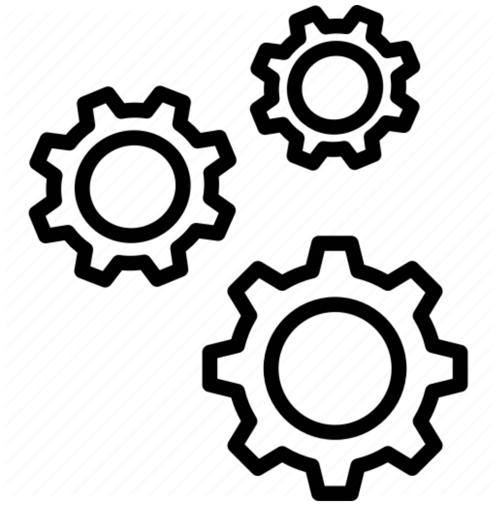
- Store key/value pairs
- Both keys and values are strings
- Can be saved to a file and loaded from a file
  - Simple text format
  - Usually go under the user's home directory
  - On UNIX systems directory name customary begins with a dot (.) to indicate a hidden system directory
- A secondary table for default values
- Simple tables without a hierarchical structure but a fake hierarchy can be introduced by naming properties appropriately. E.g., window.main.color, window.main.title, etc.

Property Maps: Disadvantages

- No concept of a home directory in some operating systems
- No standard convention for naming configuration files or their placement
- Simple text format
- Values can only be strings
- No cascading of configuration information

Preferences API

- Uses central repository of OS for storing configuration information
  - Registry in Microsoft Windows
  - Local files in UNIX
- Tree structure
  - Multiple parallel trees (user, system, etc.)
  - Hierarchical key names
- Able to export the preferences of a subtree
- Uses XML format
- Strongly typed values
  - Numbers
  - Boolean
  - String
  - Byte array



XML

- eXtensible Markup Language
- W3C standard
- An extension of SGML (Standard Generalized Markup Language)
- Data + metadata
- Data stored in attributes and inside tags
- Typically, an element:
  - Contains nested elements
  - Or contains data
- Mixed content is often discouraged

XML and HTML

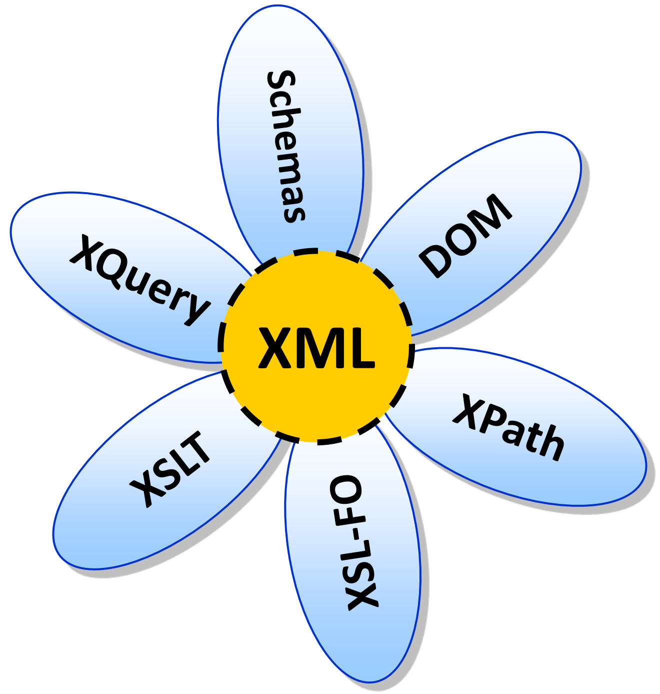
- Similar
  - Markup languages (extensions of SGML)
  - Use tags
  - Use attributes and nested tags
  - W3C standards
  - Text only, not binary data
- Different
  - Tags and attributes are predefined (HTML); tags and attributes can be anything (XML)
  - Tolerates sloppy syntax (HTML); requires that all syntax rules are strictly followed
  - Contains data and describes its presentation (HTML); only contains data (XML)

XML in Applications

- XML in initialization files
- XML as import and export format
- XML in the presentation layer
- XML as a pervasive system file format

XML Ecosystem

- Schemas – validation
- DOM – searching and editing
- XPath – addressing
- XSL-FO – formatting
- XSLT – transforming
- XQuery – querying



Applications and XML Processors

- An XML processor is a software module specially written to handle XML and to implement XML technologies
- XML processors are also called parsers
- Applications that consume or generate XML rely on XML processors
- The XML processor is a dependency in a software development project

Anatomy of XML

Processing Instructions (PI)

Elements

- **Root element**
- **Child elements**

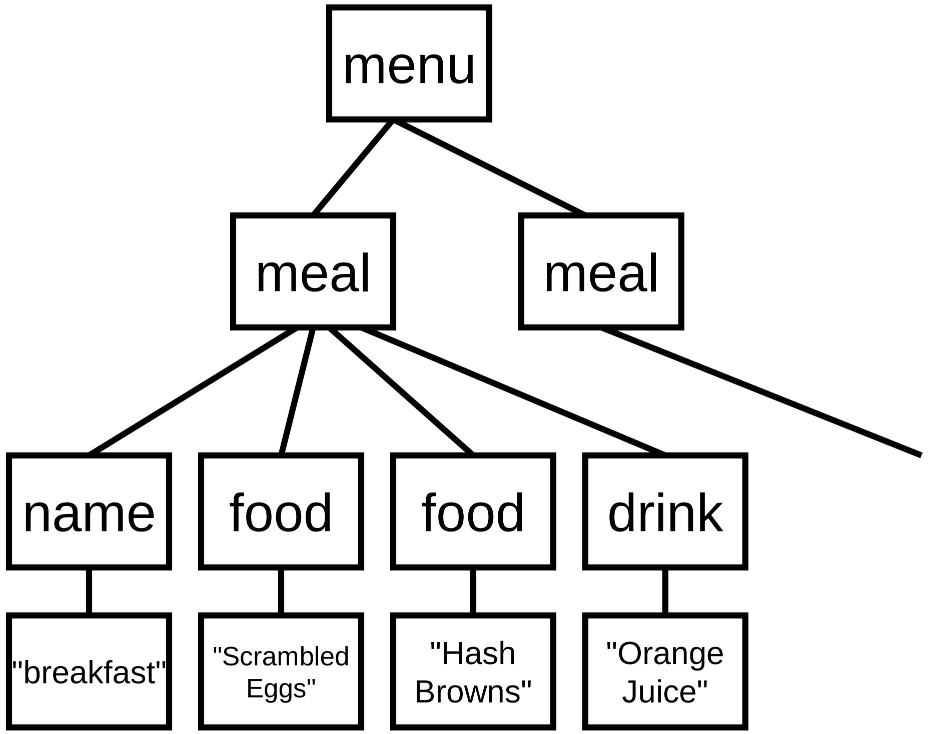
Attributes

Comments

```
<?xml version="1.0"?>
<planets>
  <planet ID="1">
    <name>Mercury</name>
  </planet>
  <planet ID="2">
    <name>Venus</name>
  </planet>
<!-- There are more planets. -->
</planets>
```

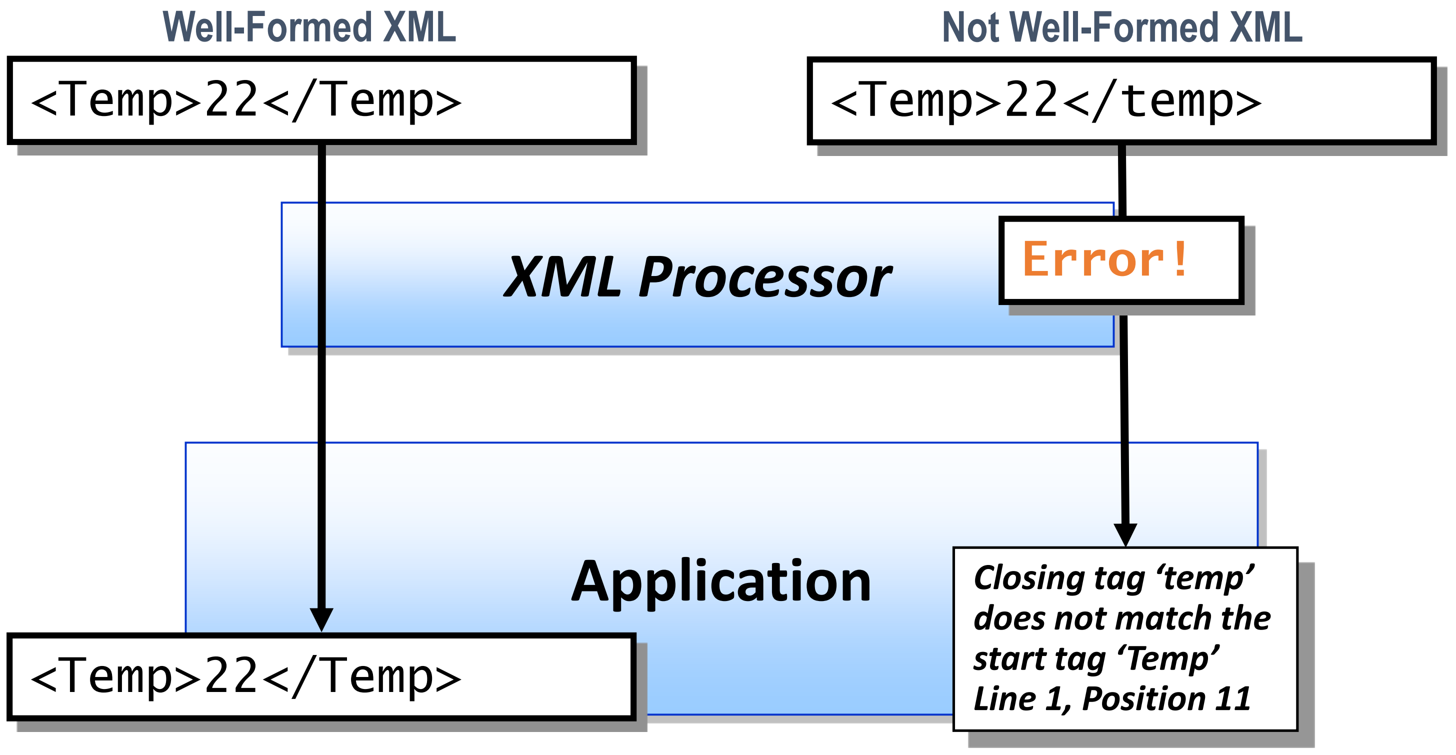
XML Is a Tree

```
<?xml version="1.0"?>
<!DOCTYPE menu SYSTEM "menu.dtd">
<menu>
   <meal name="breakfast">
            <food>Scrambled Eggs</food>
            <food>Hash Browns</food>
            <drink>Orange Juice</drink>
   </meal>
   <meal name="snack">
            <food>Chips</food>
   </meal>
</menu>
```



Well-Formed XML

- XML document is well-formed (correct) when it conforms to the specification
- An XML error stops the XML processor



Rules for Elements

```
<ElementName>element content</ElementName>
```

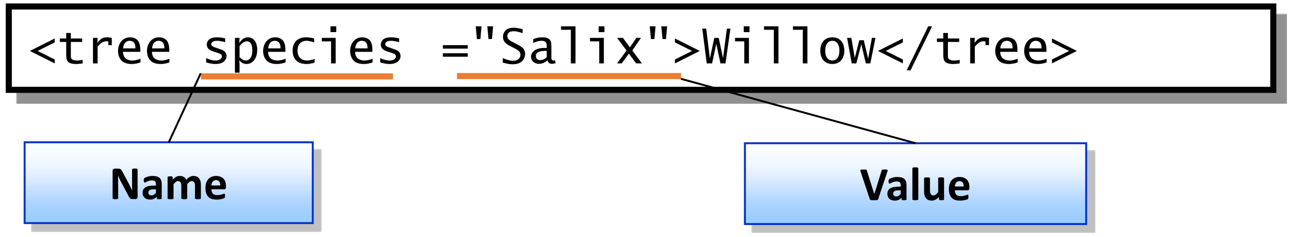
- Element names cannot contain white space
- Names cannot start with a number or a punctuation
- Names cannot start with *xml* or variants
- No space after the left angle bracket (<)
- The case of start and closing tags must match (all names are case-sensitive)
- The first element is the root element
- The root element must have start and closing tags
- All child elements must nest within the root
- Nested elements cannot overlap
- An empty child element can consist of a single tag

```
<Root>
 <ChildA>
  <ChildB>content
  </ChildB>
 </ChildA>
</Root>
```

```
<ElementName />
```

Rules for Attributes

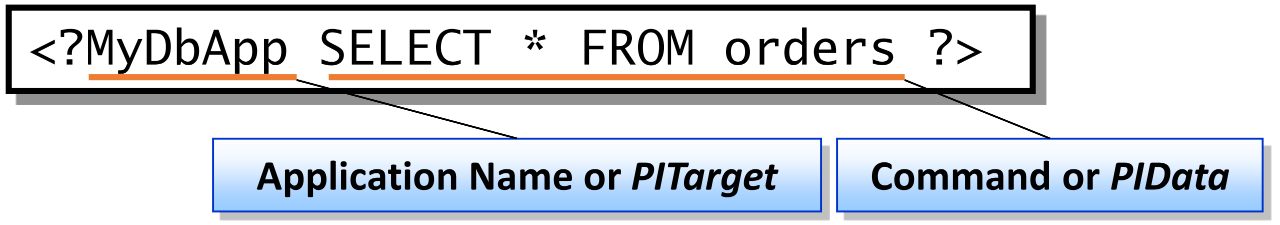
- Declare them in start tags and processing instructions
- Separate multiple declarations with a space
- An attribute consists of a name and an assignment
  - Each name must be unique within an element
  - You can reuse names throughout a document
  - There are no spaces in names
  - Use either single or double quotes for assignments



```
<tree species ="Salix">Willow</tree>
```

Processing Instructions

- Processing instructions to applications
  - Use it to send a command to an external application
  - Multiple PIs to applications are allowed



```
<?MyDbApp SELECT * FROM orders ?>
```

- Processing instructions to the XML processor – the XML declaration
  - It begins with the XML keyword
  - It must appear only once per document at the top of the document
  - Use it to declare the XML version and character encoding

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

Comments

- Do not embed a comment within a tag

Well-Formed	Not Well-Formed
`<plants><!--native --></plants>`	`<plants<!--native -->></plants>`

- Use the double hyphen (--) only to open and close the comment

`<!--native –frost tolerant -->`	`<!--native -- frost tolerant-->`

- Do not use a triple hyphen (---) at the end

`<!---10 Centigrade -->`	`<!---10 Centigrade --->`

How to Handle Reserved Characters

- Use entity references to represent reserved characters

```
<weather>Sunny & >32</weather>
```

Incorrect

```
<weather>Sunny &amp; &gt; 32</weather>
```

Correct

- Use a CDATA section to contain a block of characters

```
<![CDATA
  [SELECT Region.name, Region.location_code
FROM Region, Temp, Condition
WHERE Temp.AvgHi > 32
  AND Condition.Description = Sunny]]>
```

//TODO before next lecture:

- Homework 3 due on 3/23 at 11:59 pm EDT. Must be submitted on Submitty.
- Final Project team formation due on 3/19 at 11:59 pm EDT. Teams must be declared on Submitty.