



# Computer Science E-75

## Building Dynamic, Scalable Websites

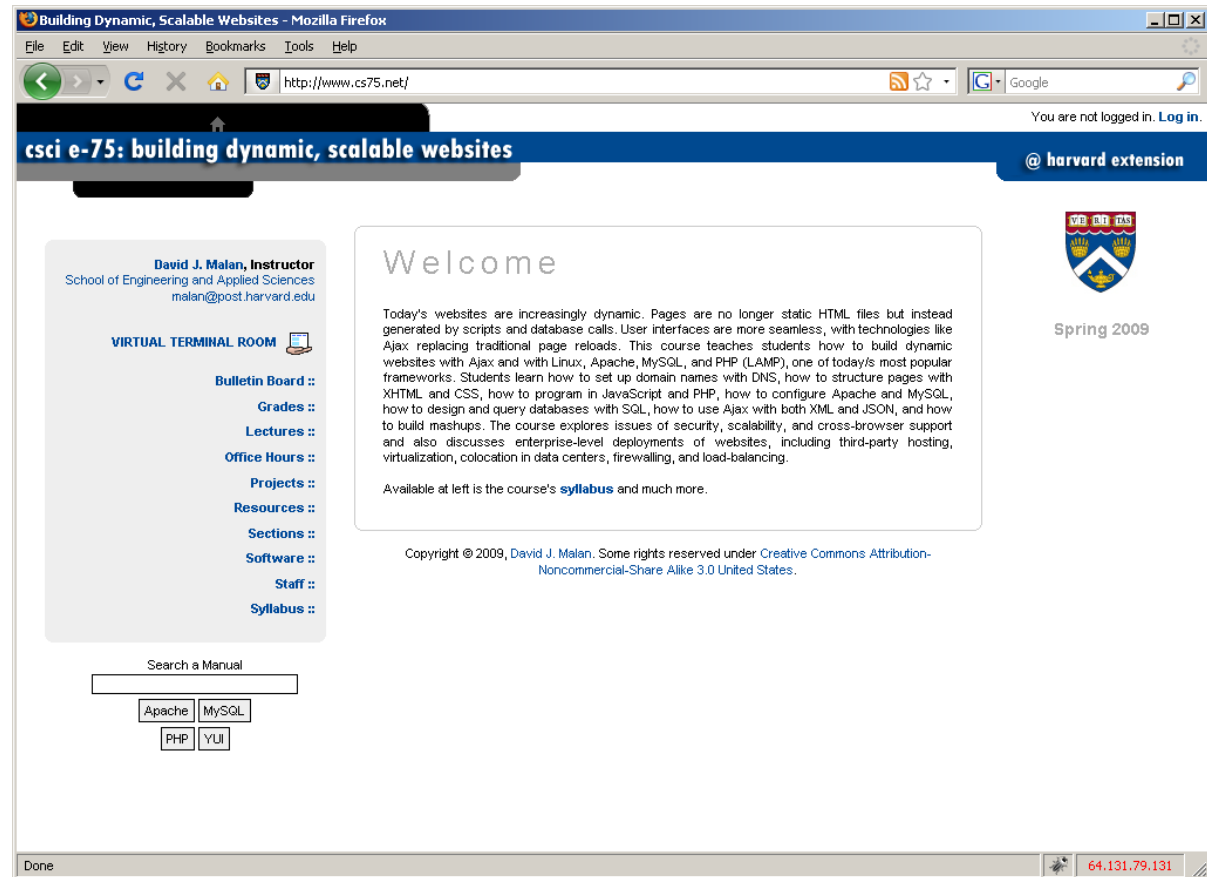
Harvard Extension School

<http://www.cs75.net/>

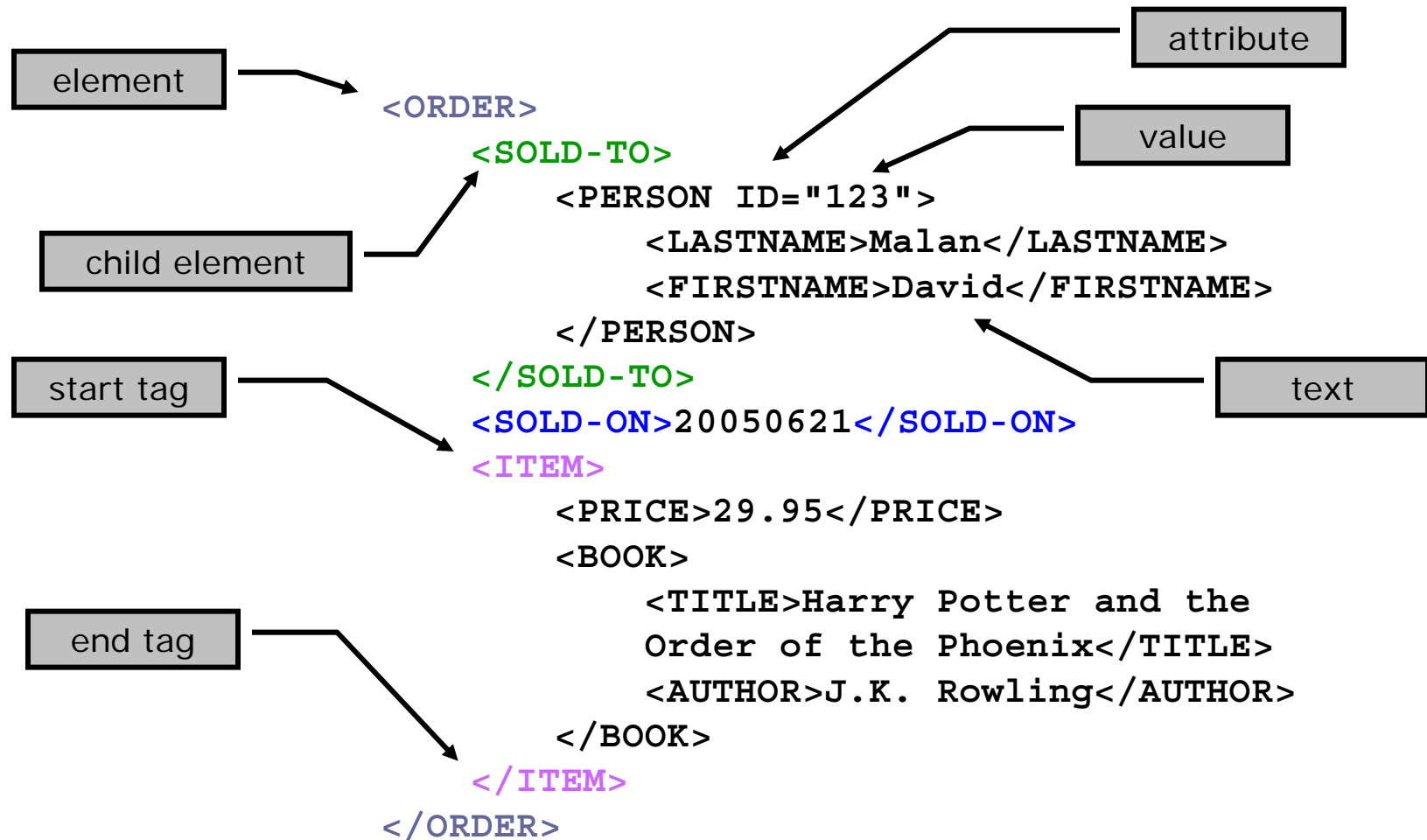
### Lecture 3: XML

David J. Malan  
[malan@post.harvard.edu](mailto:malan@post.harvard.edu)

# A Dynamic Website

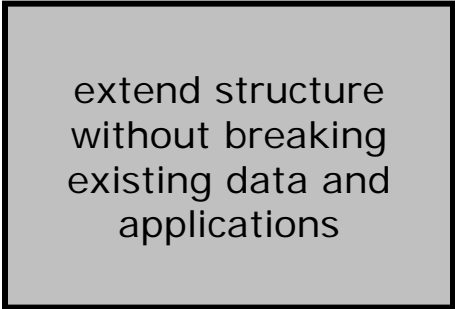


# XML



# Extensibility

```
<ORDER>
  <SOLD-TO>
    <PERSON ID="123">
      <LASTNAME>Malan</LASTNAME>
      <FIRSTNAME>David</FIRSTNAME>
      <INITIAL>J</INITIAL>
      <ADDRESS>
        <STREET>Oxford Street</STREET>
        <NUMBER>33</NUMBER>
        <CITY>Cambridge</CITY>
        <STATE>MA</STATE>
      </ADDRESS>
    </PERSON>
  </SOLD-TO>
  <SOLD-ON>20050621</SOLD-ON>
  <ITEM>
    ...
  </ITEM>
</ORDER>
```



extend structure  
without breaking  
existing data and  
applications

# <students/>

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

<!-- This is an XML document that describes students -->
<students>
  <student id="0001">
    <name>Jim Bob</name>
    <status>graduate</status>
    <dorm/>
    <major>Computer Science & Music</major>
    <description>
      <![CDATA[ <h1>Jim Bob!</h1>
        Hi my name is jim.  I look like
         ]]>
    </description>
  </student>
  <student id="0002">
    ...
  </student>
</students>
```

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

# XML Declaration

- Optional
- Must appear at the very top of an XML document
- Used to indicate the version of the specification to which the document conforms (and whether the document is “standalone”)
- Used to indicate the character encoding of the document
  - UTF-8
  - UTF-16
  - iso-8859-1
  - ...



```
<name>Jim Bob</name>
```

# Elements

- Main structure in an XML document
- Only one root element allowed
- Start Tag
  - Allows specification of zero or more attributes  
`<student id="0001" ...>`
- End Tag
  - Must match name, case, and nesting level of start tag  
`</student>`
- Name must start with letter or underscore and can contain only letters, numbers, hyphens, periods, and underscores

# Content Models

- Element Content

```
<student>  
  <status>...</status>  
</student>
```

- Parsed Character Data (aka PCDATA, aka Text)

```
<name>Jim Bob</name>
```

- Mixed Content

```
<name>Jim <initial>J</initial> Bob</name>
```

- No Content

```
<dorm/>
```





```
<student id="0001">
```

# Attributes

- Name

- ☐ Must start with letter or underscore and can contain only letters, numbers, hyphens, periods, and underscores

- Value

- ☐ Can be of several types, but is almost always a string
- ☐ Must be quoted
  - `title="Lecture 2"`
  - `match='item="baseball bat"'`
- ☐ Cannot contain `<` or `&` (by itself)



Jim Bob

# PCDATA

- Text that appears as the content of an element
- Can reference entities
- Cannot contain < or & (by itself)



`&`

# Entities

- Used to “escape” content or include content that is hard to enter or repeated frequently
  - Somewhat like macros
- Five pre-defined entities
  - `&`; `<`; `>`; `'`; `"`;
- Character entities can refer to a single character by unicode number
  - e.g., `&#x00A9;` is ©
- Must be declared to be legal
  - `<!ENTITY nbsp "&#160;">`
- Cannot refer to themselves

# CDATA

```
<![CDATA[ <h1>Jim Bob!</h1> ... ]]>
```

- Parsed in “one chunk” by the XML parser
- Data within is not checked for subelements, entities, *etc.*
- Allows you to include badly formed markup or character data that would cause a problem during parsing
- Example
  - Including HTML tags in an XML document



```
<!-- This is ... -->
```

# Comments

- Can include any text inside a comment to make it easier for human readers to understand your document
- Generally not available to applications reading the document
- Always begin with `<!--` and end with `-->`
- Cannot contain `--`



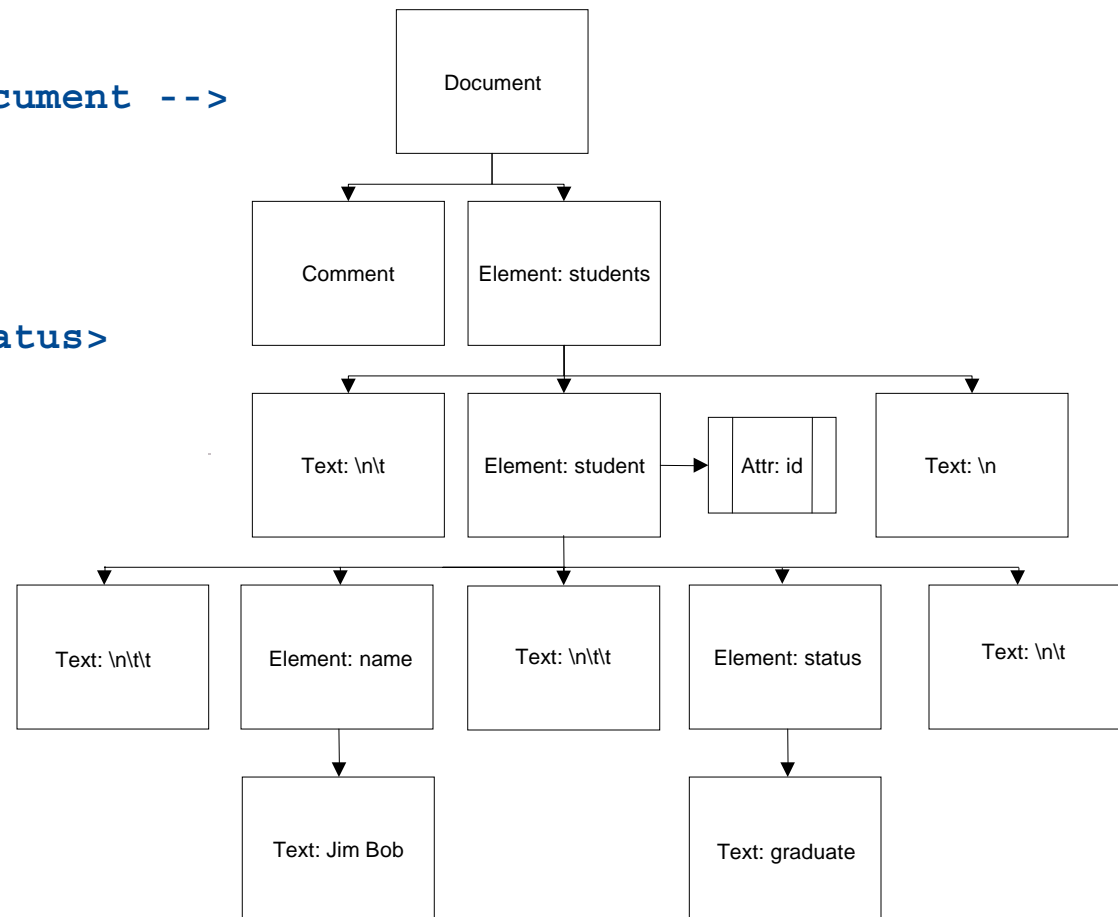
# SimpleXML

`http://us2.php.net/simplexml`

# DOM

```
<!-- smaller, simpler document -->
```

```
<students>  
  <student id="1">  
    <name>Jim Bob</name>  
    <status>graduate</status>  
  </student>  
</students>
```



# RSS

<http://cyber.law.harvard.edu/rss/rss.html>





# RSS

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title></title>
    <description></description>
    <link></link>
    <item>
      <guid></guid>
      <title></title>
      <link></link>
      <description></description>
      <category></category>
      <pubDate></pubDate>
    </item>
    [...]
  </channel>
</rss>
```

# XPath

`/child::lectures/child::lecture[@number='0']`

The diagram illustrates the components of the XPath expression `/child::lectures/child::lecture[@number='0']`. It uses curly braces to group parts of the expression and labels them below:

- A brace under `child::lectures` is labeled `step`.
- A brace under `child::` is labeled `axis`.
- A brace under `lecture` is labeled `node test`.
- A brace under `[@number='0']` is labeled `predicate`.
- A large brace under the entire expression `/child::lectures/child::lecture[@number='0']` is labeled `location path`.

# PizzaML



Image from [junkfoodnews.net](http://junkfoodnews.net).



# Computer Science E-75

## Building Dynamic, Scalable Websites

Harvard Extension School

<http://www.cs75.net/>

### Lecture 3: XML

David J. Malan  
[malan@post.harvard.edu](mailto:malan@post.harvard.edu)