core WEB programming

Introduction to XML
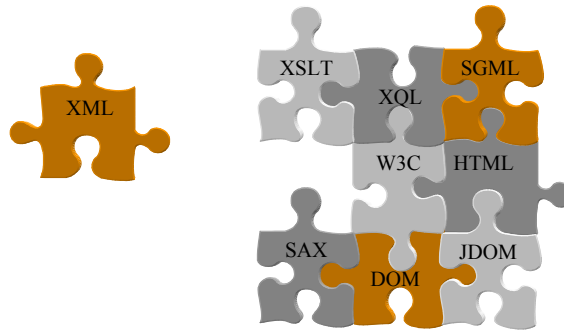

Agenda

- **XML overview**
- **XML components**
- **Document Type Definition**
- **Specifying data elements (tags)**
- **Defining attributes and entities**
- **A look at XML schema**

XML Overview

- **When people refer to XML, they typically are referring to XML and related technologies**

XML   XSLT   XQL   SGML   W3C   HTML   SAX   DOM   JDOM

www.corewebprogramming.com

XML Resources

- **XML 1.0 Specification**
  - http://www.w3.org/TR/REC-xml
- **WWW consortium's Home Page on XML**
  - http://www.w3.org/XML/
- **Sun Page on XML and Java**
  - http://java.sun.com/xml/
- **Apache XML Project**
  - http://xml.apache.org/
- **XML Resource Collection**
  - http://xml.coverpages.org/
- **O'Reilly XML Resource Center**
  - http://www.xml.com/

www.corewebprogramming.com

XML Overview

- **EXtensible Markup Language (XML) is a meta-language that describes the content of the document (self-describing data)**

  Java = Portable Programs
  XML = Portable Data

- **XML does not specify the tag set or grammar of the language**
  - Tag Set – markup tags that have meaning to a language processor
  - Grammar – defines correct usage of a language's tag

Applications of XML

- **Configuration files**
  - Used extensively in J2EE architectures
- **Media for data interchange**
  - A better alternative to proprietary data formats
- **B2B transactions on the Web**
  - Electronic business orders (ebXML)
  - Financial Exchange (IFX)
  - Messaging exchange (SOAP)

XML versus HTML

- **XML fundamentally separates content (data and language) from presentation; HTML specifies the presentation**
- **HTML explicitly defines a set of legal tags as well as the grammar (intended meaning)**

  <TABLE> … </TABLE>

- **XML allows any tags or grammar to be used (hence, eXtensible)**

  <BOOK> … </BOOK>

  – Note: Both are based on Standard Generalized Markup Language (SGML)

Simple XML Example

```
<?xml version="1.0"?>
<authors>
  <name>
    <firstname>Larry</firstname>
    <lastname>Brown</lastname>
  </name>
  <name>
    <firstname>Marty</firstname>
    <lastname>Hall</lastname>
  </name>
  ...
</authors>
```

XML Components

- **Prolog**
  - Defines the xml version, entity definitions, and DOCTYPE
- **Components of the document**
  - Tags and attributes
  - CDATA (character data)
  - Entities
  - Processing instructions
  - Comments

www.corewebprogramming.com

XML Prolog

- **XML Files always start with a prolog**

  `<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>`

  - The `version` of XML is required
  - The `encoding` identifies character set (default UTF-8)
  - The value `standalone` identifies if an *external* document is referenced for DTD or entity definition

  - Note: the prolog can contain entities and DTD definitions

www.corewebprogramming.com

Prolog Example

```
<?xml version="1.0" standalone="yes"?>
<DOCTYPE authors [
<!ELEMENT authors (name)*>
<!ELEMENT name (firstname, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
]>
<authors>
  <name>
    <firstname>James</firstname>
    <lastname>Gosling</lastname>
  </name>
  ...
</authors>
```

XML DOCTYPE

- **Document Type Declarations**
  - Specifies the location of the DTD defining the syntax and structure of elements in the document
  - Common forms:

    ```
    <!DOCTYPE root [DTD]>
    <!DOCTYPE root SYSTEM URL>
    <!DOCTYPE root PUBLIC FPI-identifier URL>
    ```

  - The root identifies the starting element (root element) of the document
  - The DTD can be external to the XML document, referenced by a SYSTEM or PUBLIC URL
    - SYSTEM URL refers to a private DTD
      - Located on the local file system or HTTP server
    - PUBLIC URL refers to a DTD intended for public use

DOCTYPE Examples

```
<!DOCTYPE book "DTDs/CWP.dtd">
```

Book must be the root element
of the XML document

DTD located in subdirectory
below XML document

```
<!DOCTYPE book SYSTEM
   "http://www.corewebprogramming.com/DTDs/CWP.dtd">
```

DTD located HTTP server:
www.corewebprogramming.com

XML DOCTYPE, cont.

- **Specifying a PUBLIC DTD**

  ```
  <!DOCTYPE root PUBLIC FPI-identifier URL>
  ```

  – The Formal Public Identifier (FPI) has four parts:
    1. Connection of DTD to a formal standard
       – if defining yourself
       + nonstandards body has approved the DTD
       ISO if approved by formal standards committee

    2. Group responsible for the DTD
    3. Description and type of document
    4. Language used in the DTD

PUBLIC DOCTYPE Examples

```
<!DOCTYPE Book
   PUBLIC "-//W3C//DTD XHMTL 1.0 Transistional//EN"
   "http://www.w3.org/TR?xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCYTPE CWP
   PUBLIC "-//Prentice Hall//DTD Core Series 1.0//EN"
   "http://www.prenticehall.com/DTD/Core.dtd">
```

XML Comments

- **Comments are the same as HTML comments**

```
<!-- This is an XML and HTML comment -->
```

Processing Instructions

- **Application-specific instruction to the XML processor**

  `<?processor-instruction?>`

- # Example

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xml" href="orders.xsl" ?>
<orders>
  <order>
    <count>37</count>
    <price>49.99</price>
    <book>
      <isbn>0130897930</isbn>
      <title>Core Web Programming Second Edition</title>
      <authors>
        <author>Marty Hall</author>
        <author>Larry Brown</author>
      </authors>
    </book>
  </order>
</orders>
```

XML Root Element

- **Required for XML-aware applications to recognize beginning and end of document**
- # Example

```
<?xml version="1.0" ?>
<book>
  <title>Core Web Programming</title>
  <contents>
    <chapter number="1">
      Designing Web Pages with HTML
    </chapter>
    <chapter number="2">
      Block-level Elements in HTML 4.0
    </chapter>
    <chapter number="3">
      Text-level Elements in HTML 4.0
    </chapter>
    ...
  </contents>
</book>
```

XML Tags

- **Tag names:**
  - Case sensitive
  - Start with a letter or underscore
  - After first character, numbers, − and . are allowed
  - Cannot contain whitespaces
  - Avoid use of colon expect for indicating namespaces
- **For a well-formed XML documents**
  - Every tag must have an end tag
    ```
    <elementOne> … </elementOne>
    ```
  - All tags are completely nested (tag order cannot be mixed)

XML Tags, cont.

- **Tags can also have attributes**

```
<message to="Gates@microsoft.com" from="Gosling@sun.com">
  <priority/>
  <text>We put the . in .com.
       What did you do?
  </text>
</message>
```

XML Attributes

- **Element Attributes**
  - Attributes provide metadata for the element
  - Every attribute must be enclosed in "" with no commas in between
  - Same naming conventions as elements

Document Entities

- **Entities refer to a data item, typically text**
  - General entity references start with & and end with ;
  - The entity reference is replaced by it's true value when parsed
  - The characters < > & ' " require entity references to avoid conflicts with the XML application (parser)

        <   >   &   "   '

- Entities are user definable

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE book [
<!ELEMENT book (title)>
<!ELEMENT title (#PCDATA)>
<!ENTITY COPYRIGHT "2001, Prentice Hall">
]>
<book>
  <title>Core Web Programming, &COPYRIGHT;</title>
</book>
```

Document Entities (Aside)

- CDATA (character data) is not parsed

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<server>
  <port status="accept">
    <![CDATA[8001 <= port < 9000]]>
  </port>
</server>
```

Well-Formed versus Valid

- **An XML document can be *well-formed* if it follows basic syntax rules**
- **An XML document is *valid* if its structure matches a Document Type Definition (DTD)**

Document Type Definition (DTD)

- **Defines Structure of the Document**
  - Allowable tags and their attributes
  - Attribute values constraints
  - Nesting of tags
  - Number of occurrences for tags
  - Entity definitions

DTD Example

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT perennials (daylily)*>
<!ELEMENT daylily (cultivar, award*, bloom, cost)+>
<!ATTLIST daylily
   status (in-stock | limited | sold-out) #REQUIRED>
<!ELEMENT cultivar (#PCDATA)>
<!ELEMENT award (name, year)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name note CDATA #IMPLIED>
<!ELEMENT year (#PCDATA)>
<!ELEMENT bloom (#PCDATA)>
<!ATTLIST bloom code (E | EM | M | ML | L | E-L) #REQUIRED>
<!ELEMENT cost (#PCDATA)>
<!ATTLIST cost discount CDATA #IMPLIED>
<!ATTLIST cost currency (US | UK | CAN) "US">
```

Defining Elements

- **<!ELEMENT name definition/type>**

  ```
  <!ELEMENT daylily (cultivar, award*, bloom, cost)+>
  <!ELEMENT cultivar (#PCDATA)>
  <!ELEMENT id (#PCDATA | catalog_id)>
  ```

- **Types**
  - ANY   Any well-formed XML data
  - EMPTY   Element cannot contain any text or child elements
  - PCDATA   Character data only (should not contain markup)
  - elements   List of legal child elements (no character data)
  - mixed   May contain character data and/or child elements
                (cannot constrain order and number of child elements)

www.corewebprogramming.com

Defining Elements, cont.

- **Cardinality**
  - [none]   Default (one and only one instance)
  - ?   0, 1
  - *   0, 1, …, N
  - +   1, 2, …, N

- **List Operators**
  - ,   Sequence (in order)
  - | Choice (one of several)

www.corewebprogramming.com

Grouping Elements

- **Set of elements can be grouped within parentheses**
  - (Elem1?, Elem2?)+
    - Elem1 can occur 0 or 1 times followed by 0 or 1 occurrences of Elem2
    - The group (sequence) must occur 1 or more times

- **OR**
  - ((Elem1, Elem2) | Elem3)*
    - Either the group of Elem1, Elem2 is present (in order) or Elem3 is present, 0 or more times

---

Element Example

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE Person [
<!ELEMENT Person ( (Mr|Ms|Miss)?, FirstName,
         MiddleName*, LastName, (Jr|Sr)? )>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT MiddleName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT Mr EMPTY>
<!ELEMENT Ms EMPTY>
...
<!ELEMENT Sr EMPTY>
]>
<Person>
  <Mr/>
  <FirstName>Lawrence</FirstName>
  <LastName>Brown</LastName>
</Person>
```

Defining Attributes

- **<!ATTLIST element attrName type modifier>**

- **Examples**

  ```
  <!ELEMENT Customer  (#PCDATA )>
  <!ATTLIST Customer id CDATA  #IMPLIED>

  <!ELEMENT Product (#PCDATA )>
  <!ATTLIST Product
            cost CDATA  #FIXED "200"
            id   CDATA  #REQUIRED>
  ```

Attribute Types

- **CDATA**
  - Essentially anything; simply unparsed data
    ```
    <!ATTLIST Customer id CDATA  #IMPLIED>
    ```
- **Enumeration**
  - attribute (value1|value2|value3) [Modifier]

- **Eight other attribute types**
  - ID, IDREF, NMTOKEN, NMTOKENS, ENTITY, ENTITIES, NOTATION

Attribute Modifiers

- **#IMPLIED**
  - Attribute is not required
    - `<!ATTLIST cost discount CDATA #IMPLIED>`
- **#REQUIRED**
  - Attribute must be present
    - `<!ATTLIST account balance CDATA #REQUIRED>`
- **#FIXED "*value*"**
  - Attribute is present and always has this value
    - `<!ATTLIST interpreter language CDATA #FIXED "EN">`

- **Default value (applies to enumeration)**
  - `<!ATTLIST car color (red | white | blue) "white" )`

Defining Entities

- **`<!ENTITY name "replacement">`**

  ```
  <!ENTITY &amp; "&">
  <!ENTITY copyright "Copyright 2001">
  ```

Limitations of DTDs

- **DTD itself is not in XML format – more work for parsers**
- **Does not express data types (weak data typing)**
- **No namespace support**
- **Document can override external DTD definitions**
- **No DOM support**

- **XML Schema is intended to resolve these issues but … DTDs are going to be around for a while**

www.corewebprogramming.com

XML Schema

- **W3C recommendation released May 2001**
  - `http://www.w3.org/TR/xmlschema-0/`
  - `http://www.w3.org/TR/xmlschema-1/`
  - `http://www.w3.org/TR/xmlschema-2/`
  - Depends on following specifications
    - XML-Infoset, XML-Namespaces, XPath
- **Benefits:**
  - Standard and user-defined data types
  - Express data types as patterns
  - Higher degree of type checking
  - Better control of occurrences

    - Clearly the future … but limited support

www.corewebprogramming.com

XML Schema, Example

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="perennials" type="PerennialType"/>

  <xsd:complexType name="PerennialType" >
    <xsd:element name=daylily" type="DaylilyType"
                 maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="DaylilyType" >
    <xsd:sequence>
      <xsd:element name="cultivar"  type="xsd:string"/>
      <xsd:element name="award"     type="AwardType"
                   maxOccurs="unbounded"/>
      <xsd:element name="bloom"     type="xsd:string"/>
      <xsd:element name="cost"      type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="status" type="StatusType"
                   use="required"/>
  </xsd:complexType>
```

XML Schema, Example, cont.

```
<xsd:simpleType name="StatusType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="in-stock"/>
    <xsd:enumeration value="limited"/>
    <xsd:enumeration value="sold-out"/>
  </xsd:restriction>
</xsd:simpleType>
  ...
</xsd:schema>
```

Summary

- **XML is a self-describing meta data**
- **DOCTYPE defines the *root* element and location of DTD**
- **Document Type Definition (DTD) defines the grammar of the document**
  - Required to *validate* the document
  - Constrains grouping and cardinality of elements
- **DTD processing is expensive**
- **Schema uses XML to specify the grammar**
  - More complex to express but easier to process

---

core

WEB

programming

Questions?