LOYOLA UNIVERSITY CHICAGO

CENTER FOR TEXTUAL STUDIES AND DIGITAL HUMANITIES

DIGH 401 - Introduction to Computing

Fall Semester 2014

Week 5

Today's Class

- Weekly Exercise
- XML
- Software Engineering (if time…)

DIGH 401 - Introduction to Computing

Weekly Exercise

Using the global scope example on slide 13 of this week's class notes, please write a PHP function to calculate the total number of combined days in the months March, April, July, and August. Then pass this result as a parameter to a new function, and minus the number of days in the month of December. Then output this new total as your final result.

DIGH 401 - Introduction to Computing

Weekly Exercise - Example Solution 1

```
<?php
$march = 31;
$april = 30;
$july = 31;
$august = 31;
$december = 31;
$subtotal;

function addDays() {
global $march, $april, $july, $august, $subtotal;
$subtotal = $march + $april + $july + $august;
}

function minusDays($days) {
global $subtotal, $december;
$subtotal = $days - $december;
}

//run function to calculate adding days
addDays();

//output first subtotal to check addition function
echo '<p>first subtotal = '.$subtotal.'</p>';

//run function to calculate subtracting days
minusDays($subtotal);

//output result of calculations
echo '<p>final subtotal = '.$subtotal.'</p>';
?>
```

- how can we improve this solution?
- any limitations?
- abstraction issues?

Demo

DIGH 401 - Introduction to Computing

Weekly Exercise - Example Solution 2

```
<?php
$subtotal;

function addDays() {
global $subtotal;
$march = 31;
$april = 30;
$july = 31;
$august = 31;
$subtotal = $march + $april + $july + $august;
}

function minusDays($days) {
global $subtotal;
$december = 31;
$subtotal = $days - $december;
}

/**run function to calculate adding days**/
addDays();

/**output first subtotal to check addition function**/
echo '<p>first subtotal = '.$subtotal.'</p>';

/**run function to calculate subtracting days**/
minusDays($subtotal);

/**output result of calculations**/
echo '<p>final subtotal = '.$subtotal.'</p>';
?>
```

- what's wrong with this solution?
- what are its limitations?
- abstraction?

Demo

Introduction to XML

- XML = EXtensible Markup Language
- markup language similar to HTML
- designed to carry data but not display data
- XML tags are not pre-defined, you must define your own
- designed to be self-descriptive
- XML is a W3C recommendation

Introduction to XML

```
<calendar>
<date>23rd September 2014</date>
<title>DIGH 401</title>
<location>CTSDH, Loyola Hall...</location>
</calendar>
```

Introduction to XML

- separation of data from display
- simplifies data sharing and transport
- simplifies and eases transition to new platforms and upgrades
- XML becomes more accessible
- XML has also been used to create new languages

XML Structure

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<calendar type="personal">
<date>20th September 2012</date>
<title>DIGH 401</title>
<location>Room 201 Cudahy Library</location>
</calendar>
```

XML Structure

Syntax

- there must be closing tags

```
eg: <p>a new paragraph...</p>  and not
    <p>a new paragraph...
```

- tags are case-sensitive
- elements must be properly nested

```
eg: <bold><italic>a new phrase...</italic></bold>  and not
    <bold><italic>a new phrase...</bold></italic>
```

- must have a root element
- attribute values must be quoted
- entity references

  eg: using a character such as < instead of &lt;

DIGH 401 - Introduction to Computing

XML Structure

Entity References

<	&lt;	less than
>	&gt;	greater than
&	&amp;	ampersand
'	&apos;	apostrophe
"	&quot;	quotation mark

XML Structure

Syntax

- comments

```
<!-- this is a comment in XML -->
```

XML Structure

```
<library>
  <book category="fiction">
    <title>To the Lighthouse</title>
    <author>Virginia Woolf</author>
    <year>1927</year>
  </book>
</library>
```

Elements

- everything from the start to end tag of an element, inclusive of the tags
- can contain
    - other elements
    - text
    - attributes
    - or a mix of all of the above...

XML Structure

Elements

- Naming rules
    - elements can contain letters, numbers, and other characters
    - elements cannot start with a number or punctuation character
    - elements cannot start with the letters xml (or XML, or Xml, etc)
    - elements cannot contain spaces
- no words are reserved considering the above rules
- best practice examples for element naming
- XML elements are extensible

XML Structure

Attributes

```
<book type="print">Hannibal's Footsteps</book>
```

- additional information about the element
- attribute values must be quoted
- double or single quotes
- no strict rules when to use attribute and when to use element
- some issues with attributes are:
    - cannot contain multiple values (elements can)
    - cannot contain tree structures (elements can)
    - not easily expandable (for future changes)
- attribute values for metadata

```
<book id="4024">A Good Year</book>
```

XML Tests

Think about how you might encode the following information:

- a car

- a sports team

- 2 Musical CDs including each song per album per CD (each album has 5 songs)

To the Lighthouse

(Here Mr. Carmichael, who was reading Virgil, blew out his candle.)

3

But what after all is one night? A short space, especially when the darkness dims so soon, and so soon a bird sings, a cock crows loudly, or a faint green quickens, like a turning leaf, in the hollow of the wave.

It seemed now as if, touched by human penitence and all its tool, divine goodness had parted the curtain and displayed behind it, single, distinct, **the hare erect**; the wave falling; the boat rocking, which, did we deserve them, should be

XML Viewing

- view without styling in a web browser, editor... - DEMO

- render with CSS by associating a stylesheet with a given XML file - DEMO

- transform the document using XSLT

- access, query and manipulate XML using Javascript

- parse XML and render using a language such as PHP
    - PHP & Javascript - DEMO

XML Validation

- 'Well Formed' XML = correct syntax
    - root element
    - elements must have closing tags
    - tags are case sensitive
    - elements must be properly nested
    - attribute values must be quoted
- 'Valid' XML is 'Well Formed' XML that conforms to a DTD