HTML, XML and Web Searching

HW9: Creativity in Processing

- In a couple of weeks when you make a web page and put your creativity programs on there, some people will want to show what they did to the class?
- HW 12: Another chance to be creative: Last two weeks of term. Get a pair programming partner again!

In Processing: File Export

Encoding Information: There's more!

- Bits and bytes encode the information, but that's not all
- **Tags** encode format and some structure in word processors
- **Tags** encode format and some structure in HTML
- **Tags** are one form of meta-data
- **Meta-data** is information about information

HTML and the Web

- The Web uses **http://** protocol
- Its asking for a Web page, which usually means a page expressed in **hyper-text markup language**, or **HTML**
  - **Hyper-text** refers to text containing **LINKS** that allow you to leave the linear stream of text, see something else, and return to the place you left
  - **Markup language** is a notation to describe how a published document is supposed to look: what kinds of fonts, text color, headings, images, etc.

Basics of HTML #1

- Rule 1: Content is given directly; anything that is not content is given inside of tags
- Rule 2: Tags made of < and > and used this way:
  - **Attribute&Value**
  - `<p style="color:red">This is paragraph.</p>`

Start               | Content | End
---                  |---------|------
Tag                 |         | Tag

It produces: **This is paragraph.**
- Rule 3: Tags must be paired or "self terminated"
There are great resources out there

- http://www.w3schools.com/html/default.asp

Let's do it.

Example

- Write HTML in text editor: notepad++ or TextWrangler
- The file extension is .html; show it in Firefox or your browser

Example: myfirst.html

- To put in an image (.gif, .jpg, .png), use 1 tag
  `<img src="cooking-ewan-isabel.jpg" alt="Kids Cooking"/>

- To put in a link, use 2 tags
  `<a href=http://users.soe.ucsc.edu/~maw>Prof. Walker's</a>

Basics of HTML #2

- Rule 4: An HTML file has this structure:
  `<html>
    <head><title>Name of Page</title></head>
    <body>
      Actual HTML page description goes here
    </body>
  </html>

- Rule 5: Tags must be properly nested
- Rule 6: White space is mostly ignored
- Rule 7: Attributes (style="color:red") preceded by space, name not quoted, value quoted

Basics of HTML #3

- To put in an image (.gif, .jpg, .png), use 1 tag
  `<img src="cooking-ewan-isabel.jpg" alt="Kids Cooking"/>

- To put in a link, use 2 tags
  `<a href=http://users.soe.ucsc.edu/~maw>Prof. Walker's</a>

- More on HTML (including good tutorials) at
  http://www.w3schools.com/html/default.asp
Meta-Data Describes Data

- Meta-data is data about data ... a description of what the data is
  - Knowing what the data is, as in the OED, allows us to process it better for users
  - Here’s an example: Search OED for def of “binary”
    - Without meta-data, get 8,311 hits ... which one is the definition?
    - With meta-data, get each definition in order ... how?

Meta-Data Describes Data

- Meta-data is data about data ... a description of what the data is
  - Knowing what the data is, as in the OED, allows us to process it better for users
  - Here’s an example: Search OED for def of “binary”
    - Without meta-data, get 8,311 hits ... which one is the definition?
    - With meta-data, get each definition in order ... how?
Give Meta-data Using Tags

- We have seen tags in several forms
  - OED ... custom tags created for dictionary
  - HTML ... the tags “mark up” the content so it can be displayed; it’s a simple use of tagging

Meta-data Separation

- Improving on the meta-data of HTML
- Meta-data describes what the data is, but because the tags can be distinguished from the content, it separates itself from the content – that’s good
- But HTML combines “identifying content” with “saying how to process it”, i.e. display it
- Big new idea (not part of HTML)

Meta-data Separation

- Separate the content and its tags entirely from the processing – produce a data-only file

The Advantage of Separating

- Separating the content (data) from the processing
  - You can do many different things with the content
    - Display it in multiple ways
    - Do calculations on it
  - You can maximize expertise
    - The content expert (you) puts the data together
    - The processing expert (some programmers) write the processing code based on the tags

The World of XML

- The Extensible Markup Language (XML) is a mark up language in which YOU think up the tags ... it is a self-defining language!
  - The usual rules for tags apply
    - Enclose in < and >
    - Start tag <mynewtag> and End tag </mynewtag>
    - Tags must always be matched or self-terminated
    - Tags can have attributes (think those up, too) of form attributename="valueInQuotes"
  - Use .xml as the file extension
  - Always start with “standard text” (shown later)

Example of XML

- Suppose I want to record information about my cd collection using XML, I might write:

```xml
<catalog>
  <cd>
    <title>Maggie May</title>
    <artist>Rod Stewart</artist>
    <country>UK</country>
    <company>Pickwick</company>
    <price>8.50</price>
    <year>1990</year>
  </cd>
  <cd>
    <title>When a man loves a woman</title>
    <artist>Percy Sledge</artist>
    <country>USA</country>
    <company>Atlantic</company>
    <price>8.70</price>
    <year>1987</year>
  </cd>
  …
</catalog>
```

I invent the tags; they make sense to me, and I can write a program to process such descriptions

Learning XML

- Since we think up the tags ourselves, it’s the easiest language in the world to learn, right?
- Tags can serve in three roles ...
  - Identity – tag it so you know what it is
  - Affinity – all properties of a thing should be collected together
  - <name>George Washington</name>
  - <company>Pickwick</company>
  - <year>1990</year>

The World of XML

- The Extensible Markup Language (XML) is a mark up language in which YOU think up the tags ... it is a self-defining language!
  - The usual rules for tags apply
    - Enclose in < and >
    - Start tag <mynewtag> and End tag </mynewtag>
    - Tags must always be matched or self-terminated
    - Tags can have attributes (think those up, too) of form attributename="valueInQuotes"
  - Use .xml as the file extension
  - Always start with “standard text” (shown later)
Ways To Use Tags

- **Identity** – tag it so you know what it is
  
  `<name>George Washington</name>`

- **Affinity** – all properties of a thing should be collected together
  
  `<personal>
    <name>George Washington</name>
    <height>6' 2"</height>
    <teeth>Wooden</teeth>
    <home>Mount Vernon</home>
  </personal>`

Example: Classify The Uses

- `<well>`
  - `<visits>`
  - `<sightseeing/New Zealand/sight>
    <title>Yosemite</title>
    <location>New Zealand</location>
    <picture url="NewZealand.jpg"/>
    <year>2012</year>
  </sightseeing>`

- `<sightseeing/Hawaii/sight>
  <title>Humu lil</title>
  <location>Honolulu</location>
  <picture url="Hawaii.jpg"/>
  <year>2008</year>`

- `<sightseeing/Yosemite/sight>
  <title>Yosemite Canyon</title>
  <location>Yosemite Valley</location>
  <picture url="Yosemite.jpg"/>
  <year>2011</year>`

A Diary Of Travels

- One XML database could have entries for trips
- Kinds of info:
  - Where you went
  - Year
  - Pictures
- Also need a title

Example: Building The XML File

```xml
<visits>
  <sightseeing/New Zealand/sight>
    <title>Yosemite</title>
    <location>New Zealand</location>
    <picture url="NewZealand.jpg"/>
    <year>2012</year>
  </sightseeing>

  <sightseeing/Hawaii/sight>
    <title>Humu lil</title>
    <location>Honolulu</location>
    <picture url="Hawaii.jpg"/>
    <year>2008</year>
  </sightseeing>

  <sightseeing/Yosemite/sight>
    <title>Yosemite Canyon</title>
    <location>Yosemite Valley</location>
    <picture url="Yosemite.jpg"/>
    <year>2011</year>
  </sightseeing>
</visits>
```

Ways To Use Tags (continued)

- **Collection** – enclose a group of items of the same type in a collective tag
  
  `<presidents>
    <prez num="1">George ...</prez>
    <prez num="2">John ...</prez>
    <prez num="3">Thomas ...</prez>
    <prez num="44">Barack ...</prez>
  </presidents>`

- These uses become intuitive quickly

Display XML with Browser

- We can see the structure of our XML (and check that it is well formed) by displaying it in Firefox
- Introduce an error in a tag and see the error message when you browse it.

Example: Building The XML File

```xml
<visits>
  <sightseeing/New Zealand/sight>
    <title>Yosemite</title>
    <location>New Zealand</location>
    <picture url="NewZealand.jpg"/>
    <year>2012</year>
  </sightseeing>

  <sightseeing/Hawaii/sight>
    <title>Humu lil</title>
    <location>Honolulu</location>
    <picture url="Hawaii.jpg"/>
    <year>2008</year>
  </sightseeing>

  <sightseeing/Yosemite/sight>
    <title>Yosemite Canyon</title>
    <location>Yosemite Valley</location>
    <picture url="Yosemite.jpg"/>
    <year>2011</year>
  </sightseeing>
</visits>
```
Big Picture of How This Works

- Here are the players in this application

![Diagram of players]

Example: Building The XML File

```xml
<title>VisitNewZealand</title>
<action pic="<myNewZealand.jpg>
I spent 3 month backpacking in New Zealand. This is a view near Wellington.
</action>
<year>2012</year>
</visit>

<title>VisitsTomlin</title>
<action pic="<myTomlin.jpg>
Our favorite small hotel in Kenosha.
</action>
<year>2013</year>
</visit>

<title>VisitsTomlin</title>
<action pic="<myTomlin.jpg>
Little Yosemite Canyon along the trail from Yosemite valley to Mariposa Lake.
</action>
<year>2013</year>
</visit>
```

Processing The File

- For this application, “processing” the file means displaying it on the screen
  - We use a browser to display XML tagged content
  - Here’s how ---
  - For each tag, we say what HTML we want, using "XSL"
  - Put those definitions into a file (they’re called templates)
  - Put directions at the top of the XML definition telling it where to find the definitions file
  - “Run” the XML in Firefox or other browser ... it does all the rest!

XSL: Extensible Markup Language

- XSL is “processing” markup language for XML ... and (of course) it’s written in XML
- Let’s take a look
- This is the top of the file

```
<template match="travels">
  <table width="500">
    <xsl:apply-templates/>
  </table>
</template>
```

The Rest of XSL Definition

```
<xsl:template match="visit">
  <table width="500">
    <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match="sight">
  <tr><td><h3>
    <xsl:apply-templates/>
  </h3></td></tr>
</xsl:template>
```

XSL: Extensible Markup Language

- Just looking at XSL, it seems very complicated because it mixes its own tags with HTML tags
  - Standard header text ... must be first in XSL file
  - Here’s how to say what XML tag this definition is for
  - HTML in here
  - Important tag, to be explained next
Let’s Just Do It: mytravel-v1.xsl

- First just build the skeleton of the XSL
- Say how to style each XML tag
- All that’s left is HTML

Top-level Tag: <travels>
- The “root” tag needs to style the main page

Continue For Other Tags
- <table width=“500”>
- <tr>
- <td>

Grab the Attribute Value
- When we have an attribute like <action pic="orangutan.gif”>
- we grab the value using a very odd syntax
- <img src="{@pic}"…>

Linking to XSL
- To make this all happen, we go to the XML and link it to the XSL file
- Add two “boiler plate” encoding lines (just copy)
- Make href point to the XSL file

Ta Dah!
- We have an app!
- ... open XML file with Firefox
Now, Add to Data As Needed

- The Travel Log app is done; we don’t expect to write more XSL, except to restyle something
- Just add to XML file

```
<visit>
  <sight>Glacier Bay</sight>
  <action src="glacierBay.jpg">
    One of the glaciers in Glacier Bay
  </action>
  <year>2012</year>
</visit>
```

Adding More KINDS of Data

- What about the pictures???
- Add an XML picture tag ... like HTML, it will be self-terminating; needs styling XSL, too

```
<map name="glacierBay-map.jpg" />
```

```
<xsl:template match="map">
  goes in XSL
</xsl:template>
```

```
<img src="{@name}" alt="Park Map" width="500"/>
```

Place <map> tags inside of <action> tags

More kinds of meta data all the time

- ‘The semantic web’
- An “ontology” i.e. “taxonomy” of the kinds of things there are in the world
  - People, place, thing, animal, organization, country
- New tags within web pages
- Could be put there by people
- Could be programmatically identified by text processing algorithms (like what Watson Jeopardy uses)

What are you supposed to learn?

- HTML let’s you programmatically indicate how a particular content should be displayed.
- It can be served up by any HTTP server anywhere in the world.
- Typically uses UTF-8 encoding to guarantee being able to be shown
- BUT XML separates the DATA from the PROCESSING of that data
- This is a powerful idea
- Processing => search it, display it etc