store process communicate

- store as XML
- communicate using HTML and CSS
- process with XSL
<!DOCTYPE html>
<html lang="en">
   <!--This is a comment. Comments are not displayed in the browser-->
   <head>
      <title>Recipe</title>
   </head>
   <body>
      ...
      ...
      <!--ingredients-->
      ...
      ...
   </section>
   <section>
      <!--method-->
      ...
      ...
   </section>
   </body>
</html>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Test</title>
  </head>
  <body>
    <section id="ingredients">
      ...
      ...
    </section>
    <section id="method">
      ...
    </section>
  </body>
</html>

putting the meaning back

css selectors:
#ingredients{
}
#method{
}

each id is unique within document
putting the meaning back

```html
<section id="method">
  <ol>
    <li>beat the eggs</li>
    <li>squeeze the lemons</li>
    ...
  </ol>
</section>

css selector:

```
putting the meaning back

<p class="ingredient">2 eggs</p>
<p class="ingredient">4 lemons</p>
...

css selector:

```
.ingredient{
}
p.ingredient{
}
```
putting the meaning back

<div class="ingredients">
  <p>2 eggs</p>
  <p>4 lemons</p>
  ...
</div>

css selector:

.ingredients p{
}
The basic idea is that we use xslt to extract data from xml. The url identifies a page where you can apply your xsl to an arbitrary web page (this will only work if the page you choose to scrape contains well-formed xml).
<collection>
  <recipe>
    <title>Spaghetti with meatballs</title>
    <description>
      A classic vegetarian dish.
    </description>
    <ingredient>
      <spaghetti>
        <grams>600</grams><ounces>21</ounces>
      </spaghetti>
      <minced beef>
        <grams>250</grams><ounces>8.75</ounces>
      </minced beef>
      <egg>
        <number>1</number>
      </egg>
    </ingredient>
    <method>
      <step>
        In a large bowl, mix the minced beef, egg and parsley together. Season to taste.
      </step>
      <step>
        Now take a teaspoon of mixture and, in your hand, roll it into a ball. Dust the ball in flour and put to one side. Repeat with the rest of the mixture.
      </step>
    </method>
    <time>
      <prep>less than 30 mins</prep>
      <cook>10 to 30 mins</cook>
    </time>
    <servings>4</servings>
    <image>
      <file>1.jpg</file>
      <title>Beef Mince</title>
    </image>
    <by>Gino D’Acampo</by>
    <from>Saturday Kitchen</from>
  </recipe>
</collection>

We represent the information in the recipe using XML
XML organises information in labelled trees. We draw them upside-down.
An exercise in the collection, processing and communication of information.
<xsl:template match="/">
<html>
<head>
<title>IL recipes</title>
</head>
<body>
<section class="foreword">
<h1>IL recipes</h1>
<p>An exercise in the collection, processing and communication of information.</p>
</section>
<xsl:for-each select="collection/recipe">
<section class="recipe">
<h1><xsl:value-of select="title" /></h1>
<aside><xsl:value-of select="description" /></aside>
<h2>Ingredients</h2>
<ul>
<xsl:for-each select="ingredient">
<li><xsl:value-of select="." /></li>
</xsl:for-each>
</ul>
<h2>Method</h2>
<ol>
<xsl:for-each select="method/step">
<li><xsl:value-of select="." /></li>
</xsl:for-each>
</ol>
</section>
</xsl:for-each>
</body>
</html>
</xsl:template>
<section class="foreword">
    <h1>IL recipes</h1>
    <p>An exercise in the collection, processing and communication of information.</p>
</section>

<xsl:for-each select="collection/recipe">
    <section class="recipe">
        <h1><xsl:value-of select="title" /></h1>
        <aside><xsl:value-of select="description" /></aside>
        <h2>Ingredients</h2>
        <ul>
            <xsl:for-each select="ingredient">
                <li><xsl:value-of select="." /></li>
            </xsl:for-each>
        </ul>
        <h2>Method</h2>
        <ol>
            <xsl:for-each select="method/step">
                <li><xsl:value-of select="." /></li>
            </xsl:for-each>
        </ol>
    </section>
</xsl:for-each>
<h2>Ingredients</h2>
<ul>
   <xsl:for-each select="ingredient">
      <li><xsl:value-of select="." /></li>
   </xsl:for-each>
</ul>

<h2>Method</h2>
<ol>
   <xsl:for-each select="method/step">
      <li><xsl:value-of select="." /></li>
   </xsl:for-each>
</ol>

Tuesday, 6 December 11
<html>
<head>
  <title>IL recipes</title>
</head>
<body>
  <section class="foreword">
    <h1>IL recipes</h1>
    <p>An exercise in the collection, processing and communication of information.</p>
  </section>
  <xsl:for-each select="collection/recipe">
    <section class="recipe">
      <h1><xsl:value-of select="title" /></h1>
      <aside><xsl:value-of select="description" /></aside>
      <h2>Ingredients</h2>
      <ul>
        <xsl:for-each select="ingredient">
          <li><xsl:value-of select="." /></li>
        </xsl:for-each>
      </ul>
      <h2>Method</h2>
      <ol>
        <xsl:for-each select="method/step">
          <li><xsl:value-of select="." /></li>
        </xsl:for-each>
      </ol>
    </section>
  </xsl:for-each>
</body>
</html>
<recipe>
<title>Spaghetti with meatballs</title>
<description>A classic vegetarian dish.</description>

<ingredient>spaghetti</ingredient>
<grams>600</grams><ounces>21</ounces>

<ingredient>minced beef</ingredient>
<grams>250</grams><ounces>8.75</ounces>

<ingredient>egg<number>1</number></ingredient>

<method>
<step>
In a large bowl, mix the minced beef, egg and parsley together. Season to taste.
</step>
<step>
Now take a teaspoon of mixture and, in your hand, roll it into a ball. Dust the ball in flour and put to one side. Repeat with the rest of the mixture.
</step>
...
</method>
<time>
<prep>less than 30 mins</prep>
<cook>10 to 30 mins</cook>
</time>
<servings>4</servings>

<image>
<file>1.jpg</file><title>Beef Mince</title>
</image>

<by>Gino D'Acampo</by>
<from>Saturday Kitchen</from>
</recipe>

XML represents the information.
XSL transforms it to a form suited for presentation.
CSS provides the styling.
<recipe>
  <title>Spaghetti with meatballs</title>
  <description>A classic vegetarian dish.</description>
  <ingredient>spaghetti</ingredient>
    <grams>600</grams>
    <ounces>21</ounces>
  </ingredient>
  <ingredient>minced beef</ingredient>
    <grams>250</grams>
    <ounces>8.75</ounces>
  </ingredient>
  <ingredient>egg</ingredient>
    <number>1</number>
  </ingredient>
  <method>
    <step>
      In a large bowl, mix the minced beef, egg and parsley together. Season to taste.
    </step>
    <step>
      Now take a teaspoon of mixture and, in your hand, roll it into a ball. Dust the ball in flour and put to one side. Repeat with the rest of the mixture.
    </step>
    ...
  </method>
  <time>
    <prep>less than 30 mins</prep>
    <cook>10 to 30 mins</cook>
  </time>
  <servings>4</servings>
  <image>
    <file>1.jpg</file>
    <title>Beef Mince</title>
  </image>
  <by>Gino D'Acampo</by>
  <from>Saturday Kitchen</from>
</recipe>

XML represents the information.
XSL transforms it to a form suited for presentation.
CSS provides the styling.
<recipe>
  <title>Spaghetti with meatballs</title>
  <description>A classic vegetarian dish.</description>
  <ingredient>spaghetti</ingredient>
  <grams>600</grams><ounces>21</ounces>
  <ingredient>minced beef</ingredient>
  <grams>250</grams><ounces>8.75</ounces>
  <ingredient>egg<number>1</number></ingredient>
  <method>
    <step>
      In a large bowl, mix the minced beef, egg and parsley together. Season to taste.
    </step>
    <step>
      Now take a teaspoon of mixture and, in your hand, roll it into a ball. Dust the ball in flour and put to one side. Repeat with the rest of the mixture.
    </step>
  </method>
  <time>
    <prep>less than 30 mins</prep>
    <cook>10 to 30 mins</cook>
  </time>
  <servings>4</servings>
  <image>
    <file>1.jpg</file><title>Beef Mince</title>
  </image>
  <by>Gino D'Acampo</by>
  <from>Saturday Kitchen</from>
</recipe>
Here is an example

http://www.weightlossforall.com/
<table>
<thead>
<tr>
<th>BREADS &amp; CEREALS</th>
<th>Portion size *</th>
<th>per 100 grams (3.5 oz)</th>
<th>energy content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel (1 average)</td>
<td>140 cals (45g)</td>
<td>310 cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Biscuit digestives</td>
<td>86 cals (per biscuit)</td>
<td>480 cals</td>
<td>High</td>
</tr>
<tr>
<td>Jaffa cake</td>
<td>48 cals (per biscuit)</td>
<td>370 cals</td>
<td>Med-High</td>
</tr>
<tr>
<td>Bread white (thick slice)</td>
<td>96 cals (1 slice 40g)</td>
<td>240 cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Bread wholemeal (thick)</td>
<td>88 cals (1 slice 40g)</td>
<td>220 cals</td>
<td>Low-med</td>
</tr>
<tr>
<td>Chapatis</td>
<td>250 cals</td>
<td>300 cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Cornflakes</td>
<td>130 cals (35g)</td>
<td>370 cals</td>
<td>Med-High</td>
</tr>
<tr>
<td>Crackerbread</td>
<td>17 cals per slice</td>
<td>325 cals</td>
<td>Low Calories</td>
</tr>
<tr>
<td>Cream crackers</td>
<td>35 cals (per cracker)</td>
<td>440 cals</td>
<td>Low / portion</td>
</tr>
<tr>
<td>Crumpets</td>
<td>93 cals (per crumpet)</td>
<td>198 cals</td>
<td>Low-Med</td>
</tr>
<tr>
<td>Flapjacks basic fruit mix</td>
<td>320 cals</td>
<td>500 cals</td>
<td>High</td>
</tr>
<tr>
<td>Macaroni (boiled)</td>
<td>238 cals (250g)</td>
<td>95 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Muesli</td>
<td>195 cals (50g)</td>
<td>390 cals</td>
<td>Med-high</td>
</tr>
<tr>
<td>Naan bread (normal)</td>
<td>300 cals (small plate size)</td>
<td>320 cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Noodles (boiled)</td>
<td>175 cals (250g)</td>
<td>70 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Pasta (normal boiled)</td>
<td>330 cals (300g)</td>
<td>110 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Pasta (wholemeal boiled)</td>
<td>315 cals (300g)</td>
<td>105 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Porridge oats (with water)</td>
<td>193 cals (350g)</td>
<td>55 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Potatoes** (boiled)</td>
<td>210 cals (300g)</td>
<td>70 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Potatoes** (roast)</td>
<td>420 cals (300g)</td>
<td>140 cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Rice (white boiled)</td>
<td>420 cals (300g)</td>
<td>140 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Rice (egg-fried)</td>
<td>500 cals</td>
<td>200 cals</td>
<td>High in portion</td>
</tr>
<tr>
<td>Rice (Brown)</td>
<td>405 cals (300g)</td>
<td>135 cals</td>
<td>Low calorie</td>
</tr>
<tr>
<td>Rice cakes</td>
<td>28 Cals = 1 slice</td>
<td>373 Cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Ryvita Multi grain</td>
<td>37 Cals per slice</td>
<td>331 Cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Ryvita + seed &amp; Oats</td>
<td>180 Cals 4 slices</td>
<td>362 Cals</td>
<td>Medium</td>
</tr>
<tr>
<td>Spaghetti (boiled)</td>
<td>303 cals (300g)</td>
<td>101 cals</td>
<td>Low calorie</td>
</tr>
</tbody>
</table>
In the example, we used XSL to describe a simple path from the root (/) to the table rows we are interested in. XSL provides patterns for accessing various parts of a tree starting from the current node (self).
XSL lets us construct new trees out of old trees.
Once we have identified the rows we are interested in, we add a template that specifies how we will process each row.
Using variables, we can give the table data items we have selected informative names – this will make it easier to understand our code and modify it later.
To select only some rows for processing, use `xsl:if`

Here we will only process rows where the food name contains the tag 'b'. To provide this string value to the tag variable, we put it inside double quotes, "b". `contains(s, t)` is a relation between two strings, s and t – it is true if t occurs as a substring of s.

In XSL variables are just local names for constants - they don't vary. Using variables can make your stylesheet more compact and easier to understand. However, since variables don't vary, we can't use a tag variable to provide different tags to our template. If we really want to use this template with different tag values we have to declare tag as a parameter, not a variable. We do this on the next slide.
Tuesday, 6 December 11

Here, the template for processing a table row (tr) declares tag as a parameter, which is used to select only some of the rows for processing. We can use this template just as we did before, but now 'b' is just a default value for the parameter.
To supply a different value for the parameter we use xsl:with-param when the template is called (e.g. by apply-templates).

```xml
<xsl:template match="/"
    <xsl:variable name="myTag" select="'a'" />
    <xsl:if test="$myTag != ''">
        <h2>Foods containing "<xsl:value-of select="$myTag" />"></h2>
    </xsl:if>
    <xsl:apply-templates select="html/body/table[3]//table/tr">
        <xsl:with-param name="tag" select="$myTag" />
    </xsl:apply-templates>
    <hr />
</xsl:template>

<xsl:template match="tr">
    <xsl:param name="tag" select="'b'"/>
    <xsl:variable name="food" select="string(td[1])" />
    <xsl:variable name="calories" select="string(td[3])" />
    <xsl:if test="contains($food, $tag)">
        <br /><xsl:value-of select="$food"/> :: <xsl:value-of select="$calories"/>
    </xsl:if> 
</xsl:template>
```
If no default is specified, and no value is specified by xsl:apply-templates the default default is the empty string.