a talk for ApacheCon Europe 2008 by Jeremy Quinn

Break My Site

code addendum

photo credit: Môsieur J

code samples for Break My Site

data logger

```
    final PerformanceLogger write_logger =
         this.new PerformanceLogger("write", id);
    try {
       // do the write
    } finally {
      write_logger.end();
    }
```

```
protected class PerformanceLogger {
  private final long starttime;
  private final String key;
  private final String id;

  public PerformanceLogger(final String key, final String id) {
    this.key = key;
    this.id = id;
    this.starttime = performanceLogging ? System.currentTimeMillis() : 0l;
  }

  public void end() {
    if (performanceLogging) {
      final String duration = "" + (System.currentTimeMillis() - this.starttime);
      performanceLog.info("timing." + this.key + " " + this.id + " " + duration);
      final String memory = "" + Runtime.getRuntime().freeMemory();
      performanceLog.info("memory." + this.key + " " + this.id + " " + memory);
    }
  }
}
```


a simple inner class to log different measurements
designed to wrap each step we want to capture

the data

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2008-01-24T10:37:56</date>
  <millis>1201167476103</millis>
  <sequence>3</sequence>
  <logger>app.Uploader</logger>
  <level>INFO</level>
  <class>app.Uploader$PerformanceLogger</class>
  <method>end</method>
  <thread>10</thread>
  <message>timing.read 29 453</message>
</record>
<record>
  <date>2008-01-24T10:37:56</date>
  <millis>1201167476119</millis>
  <sequence>4</sequence>
  <logger>app.Uploader</logger>
  <level>INFO</level>
  <class>app.Uploader$PerformanceLogger</class>
  <method>end</method>
  <thread>10</thread>
  <message>memory.read 29 2077528</message>
</record>
<record>
  <date>2008-01-24T10:37:58</date>
  <millis>1201167478556</millis>
```

```
<type>timing</type>
<key>read</key>
<id>29</id>
<data>453</data>
```


This is the raw XML data output by the PerformanceLogger
The first transformation step extracts values from the message

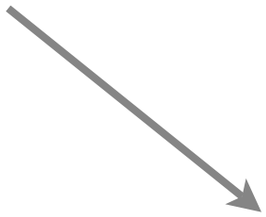
extract transformation

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" />

  <!-- don't need these, filter them out -->
  <xsl:template match="record[class != 'app.Uploader$PerformanceLogger']"/>
  <xsl:template match="date|millis|sequence|logger|level|class|method|thread"/>

  <xsl:template match="message">
    <xsl:variable name="type" select="substring-before(.,'.')"/>
    <xsl:variable name="key" select="substring-before(substring-after(.,concat($type,'.')),' ')"/>
    <xsl:variable name="id" select="substring-before(substring-after(.,' '),' ')"/>
    <xsl:variable name="data" select="substring-after(.,concat($fid,' '))"/>
    <type><xsl:value-of select="$type"/></type>
    <key><xsl:value-of select="$key"/></key>
    <id><xsl:value-of select="$id"/></id>
    <data><xsl:value-of select="$data"/></data>
  </xsl:template>

  <xsl:template match="@*|node()">
    <xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```


xslt to extract content of message into separate tags

the data

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
   <record>
      <type>timing</type>
      <key>read</key>
      <id>29</id>
      <data>453</data>
   </record>
   <record>
      <type>memory</type>
      <key>read</key>
      <id>29</id>
      <data>2077528</data>
   </record>
   <record>
      <type>timing</type>
      <key>folder</key>
      <id>29</id>
      <data>2437</data>
   </record>
   <record>
      <type>memory</type>
```



```
timing-test-1.csv :

id,read,folder,transform,write.preview,write.live,delete,
29,453,2437,1751,5437,1078,15,
16,313,1734,1297,1719,797,0,
17,47,4969,157,3094,2312,0,
69,422,1937,1110,2155,1782,0,
2,406,1734,1172,2312,860,0,
6,359,1156,1203,1390,1579,15,
25,313,2390,1344,1375,750,0,
. . .
```

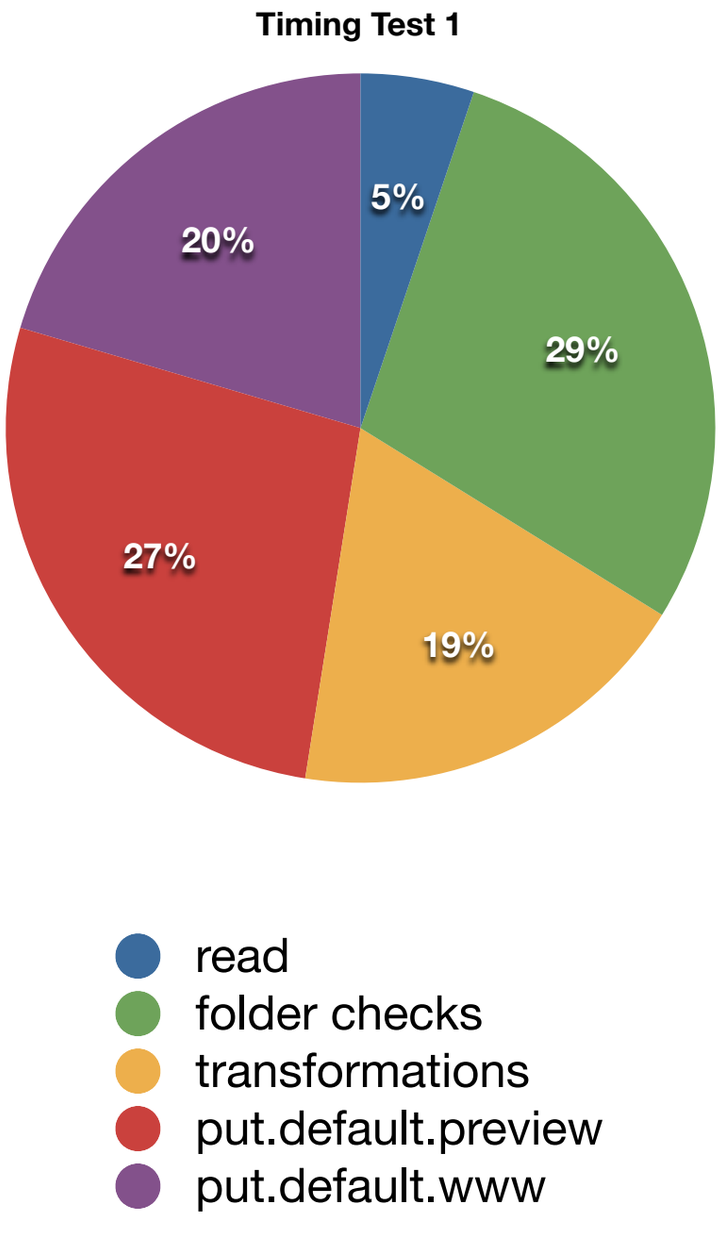

The next transformation is to convert the vast amount of flat xml
into one CSV file for each unique record type eg. 'timing' or 'memory'
where there is one row per unique record/id
where multiple values of the same record/type are summed

group transformation

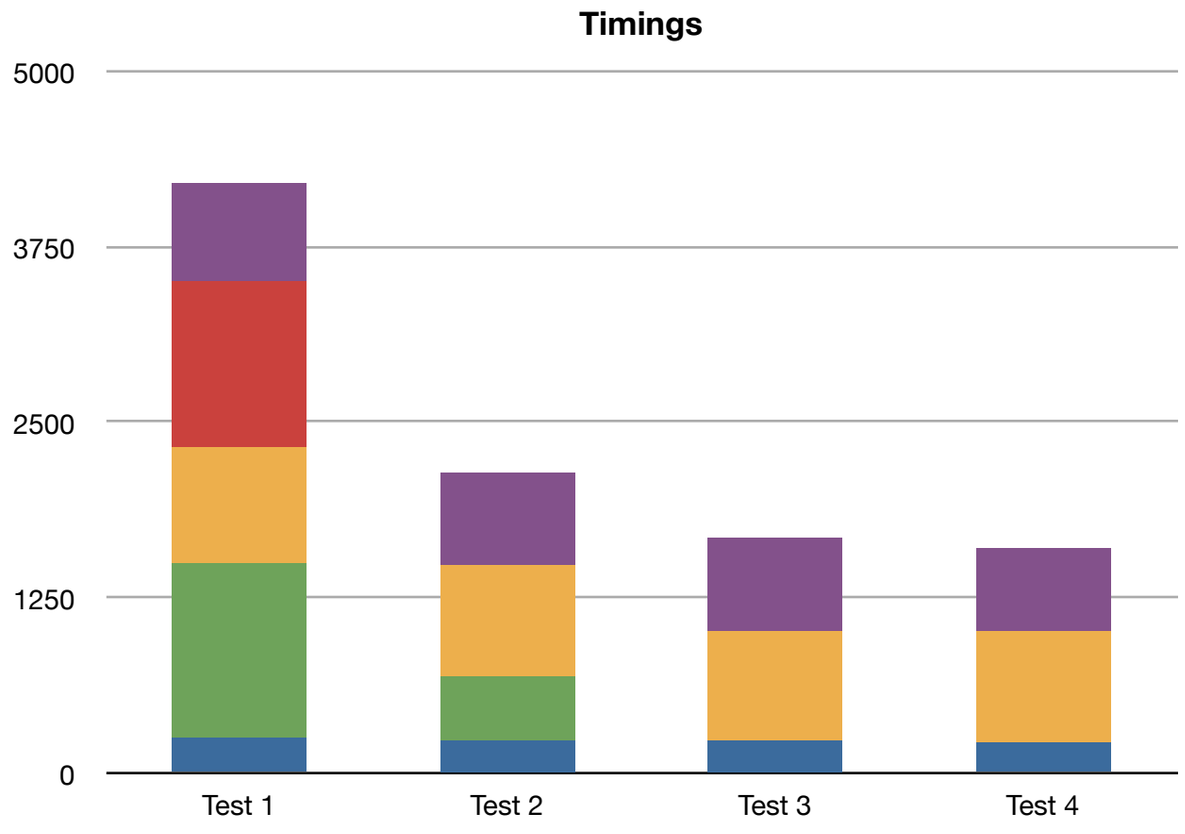
```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="test">1</xsl:param>
  <xsl:output method="text"/>

  <xsl:template match="log">
    <xsl:for-each-group select="record" group-by="type">
      <xsl:value-of select="current-grouping-key()"/><xsl:text>
</xsl:text>
      <xsl:result-document href="results/{current-grouping-key()}-{$test}.csv" method="text">
        <xsl:text>id,</xsl:text><!-- output each unique record/key into the first row -->
        <xsl:for-each-group select="current-group()" group-by="key">
          <xsl:value-of select="current-grouping-key()"/><xsl:text>,</xsl:text>
        </xsl:for-each-group>
        <xsl:text>
</xsl:text>
        <xsl:for-each-group select="current-group()" group-by="id"><!-- output one row per unique record/id -->
          <xsl:for-each select="current-group()[1]">
            <xsl:value-of select="current-grouping-key()"/><xsl:text>,</xsl:text>
            <xsl:for-each-group select="current-group()" group-by="key"><!-- output the sum of same-named keys -->
              <xsl:value-of select="sum(current-group()/data)"/><xsl:text>,</xsl:text>
            </xsl:for-each-group>
          </xsl:for-each>
          <xsl:text>
</xsl:text>
        </xsl:for-each-group>
      </xsl:result-document>
    </xsl:for-each-group>
  </xsl:template>

  <xsl:template match="@*|node()">
    <xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```


XSLT 2.0 has better grouping commands
simpler code and ran faster than XSLT 1.0

test results



- **Test 1**: the original state
- **Test 2**: write optimisation
- **Test 3**: + folder optimisation
- **Test 4**: + indexer optimisation



the CSV files are then imported into a spreadsheet and plotted as graphs