Content Transformation with Apache

Nick Kew

WebThing Limited http://www.webthing.com/

and

Apache Software Foundation http://www.apache.org/

Content Transformation

- History
- Technology Foundations
- Projects, Applications and Modules

Prehistory: Apache 1

- CGI
- PHP
- Scripting language modules
- Java/Tomcat
- Application Servers
- Advanced techs: Axkit, gd, etc.

History: Apache 2

- 2002: First production release of Apache 2
- 2002: First filter modules, XSLT, Accessibility Proxy & Content Assembly
- 2003: Modular filter applications architecture, Reverse Proxy, xmlns, ESI, WML gateway
- 2004: Accelerator Proxy, image processing proxy, smart filtering, mod_publisher, mod_annot, embedded SQL.
- 2005: Line editor. Apache 2.2.

Proxy

- Apache 1: simple all-in-one HTTP/1.0 proxy module
- Apache 2: HTTP/1.1; separate mod_proxy, mod_cache
- protocol framework (HTTP, FTP, AJP, ...)
- Load Balancing
- Apache 2.2: proxy+cache matures

Cache

History

- Apache 1: HTTP/1.0 Cache
- Apache 2: HTTP/1.1 Support
- Apache 2.2: Mature HTTP/1.1

Future (provisional): Cache Framework

- mod_cache becomes mod_cache_http
- Support other cacheing strategies such as ESI, CML, mod_carp

Apache 2: Filters

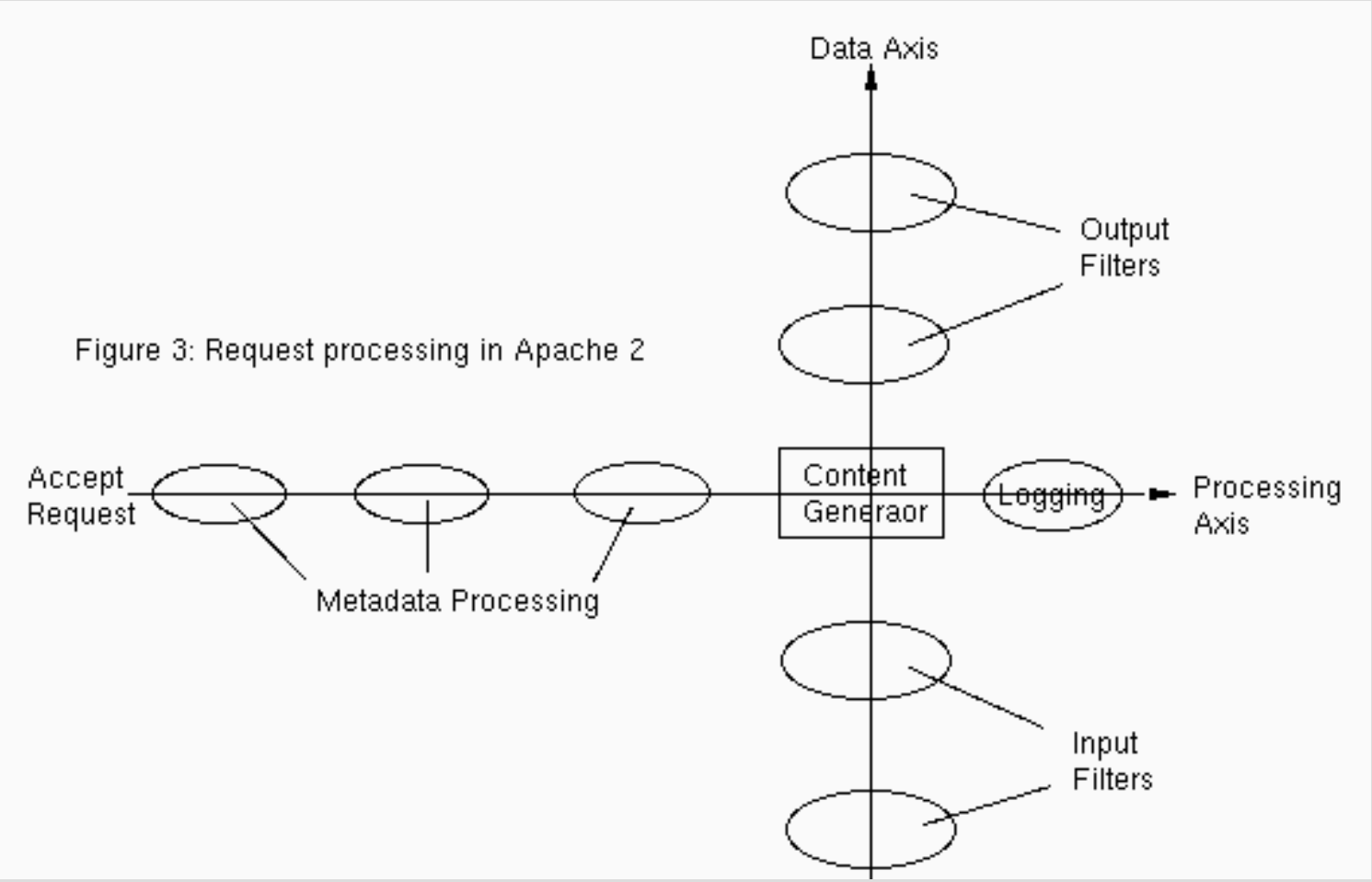
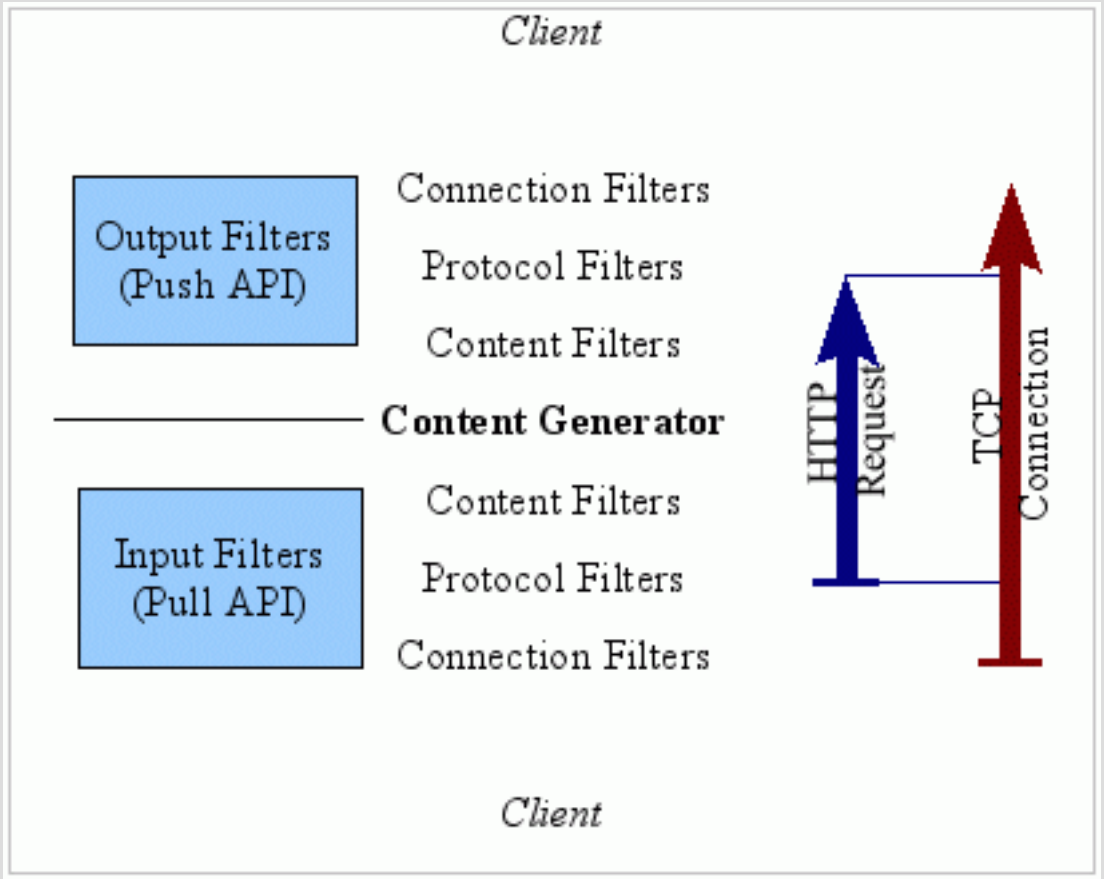


Figure 3: Request processing in Apache 2

The Data Axis



Filter Types

- Content — SSI, XML/XSLT
- Content Encoding — Compression, Charset
- Protocol — Headers, Byteranges
- Protocol Encoding — Chunking
- Connection — SSL, Bandwidth
- Network — TCP, AcceptFilter

Content transformation

- mod_includes (SSI)
- mod_charset_lite (iconv)
- mod_deflate (compression)

Many third-party modules

- XSLT filters
- mod_proxy_html
- etc

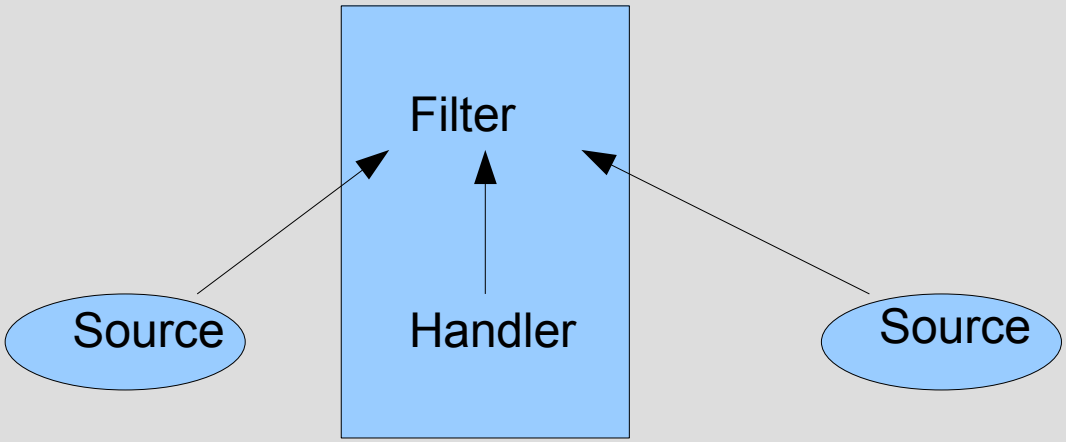
Filtering Proxy with Cache

```
<VirtualHost 192.168.65.19>
    ServerName proxy.example.com
    ProxyRequests On
    Order Deny,Allow
    Deny from All
    Allow from 192.168.65
    CacheEnable disk /
    CacheRoot /var/spool/www-cache
    FilterProvider transform accessibility $text/html
    FilterChain  unpack transform repack
</VirtualHost>
```

Reverse Proxy

```
ProxyRequests Off
ProxyPass /foo/  http://foo.example.com/
ProxyHTMLURLMap http://foo.example.com /foo
<Location /foo/>
    ProxyPassReverse /
    ProxyHTMLURLMap / /foo/
    FilterProvider transform proxy-html $text/html
    FilterChain unpack transform repack
</Location>
CacheEnable disk /foo/
CacheRoot /var/spool/www-cache
```

Content Assembly



Content Assembly

- SSI <!--#include ... -->
- ESI <esi:include ...>
- SQL <sql:select ...>
- Scripting, Templating, ...
- Data Discovery
- Aggregation
- Image Processing

Caching often crucial to performance

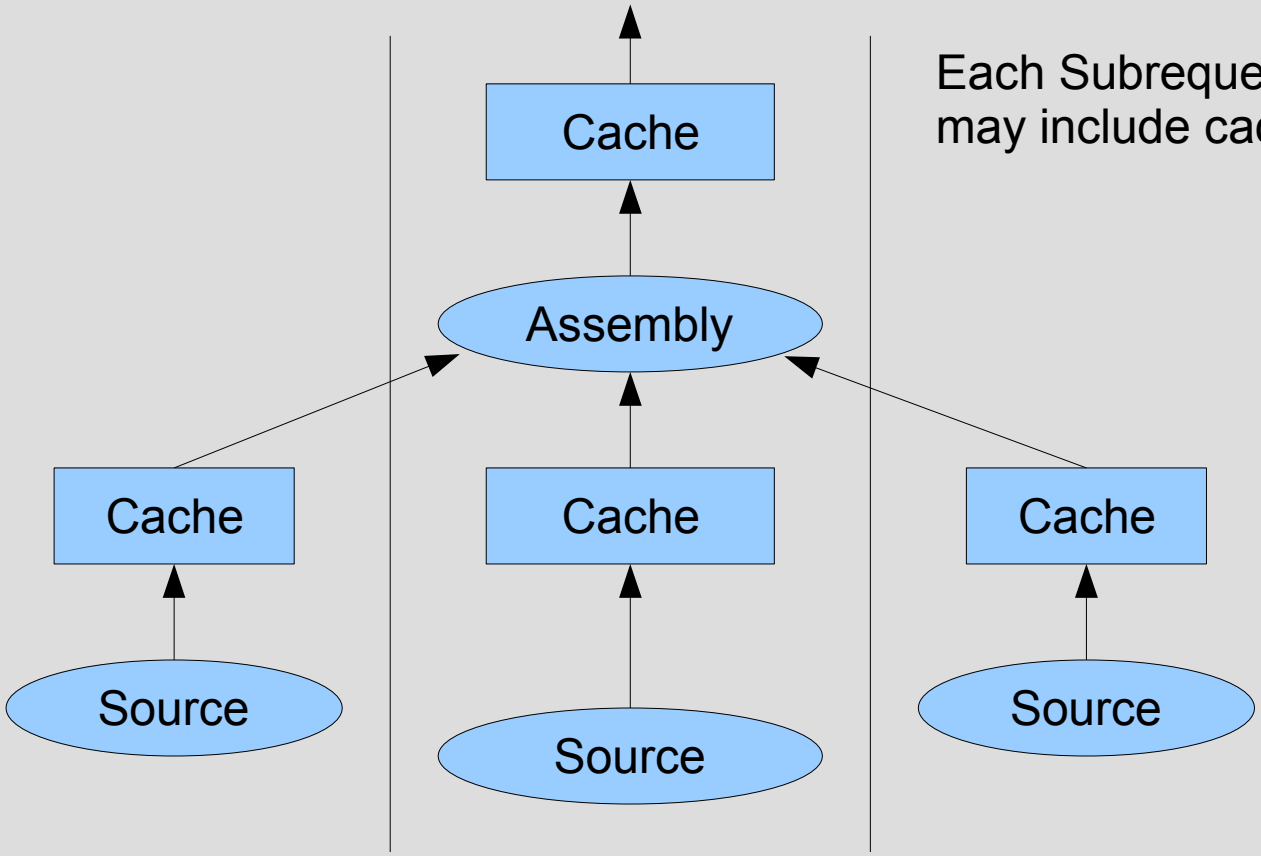
Content Assembly

At the browser:

- <img src ...>
- <object data ...>
- frame, applet, script, ...

No concern of Apache, except to ensure correct HTTP (caching, etc). But we need to consider where content assembly is best accomplished.

Multi-level Caching



Each Subrequest
may include cache

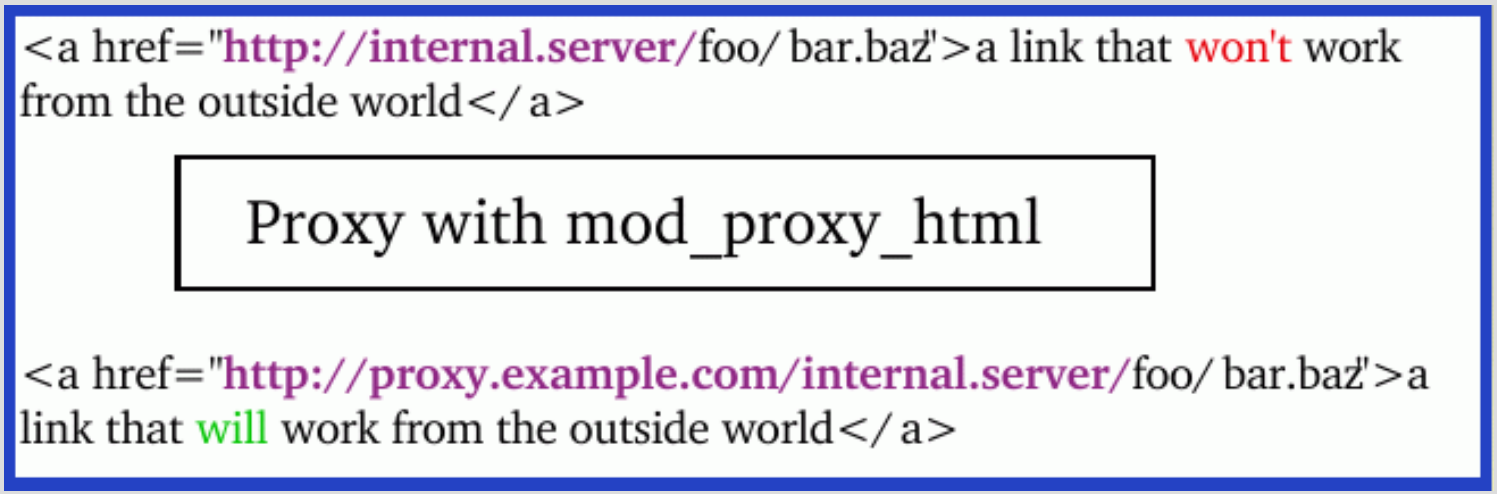
Accessibility

- BBC 'betsie' Perl script
- Accessibility Proxy (2002)
- mod_accessibility (2003)

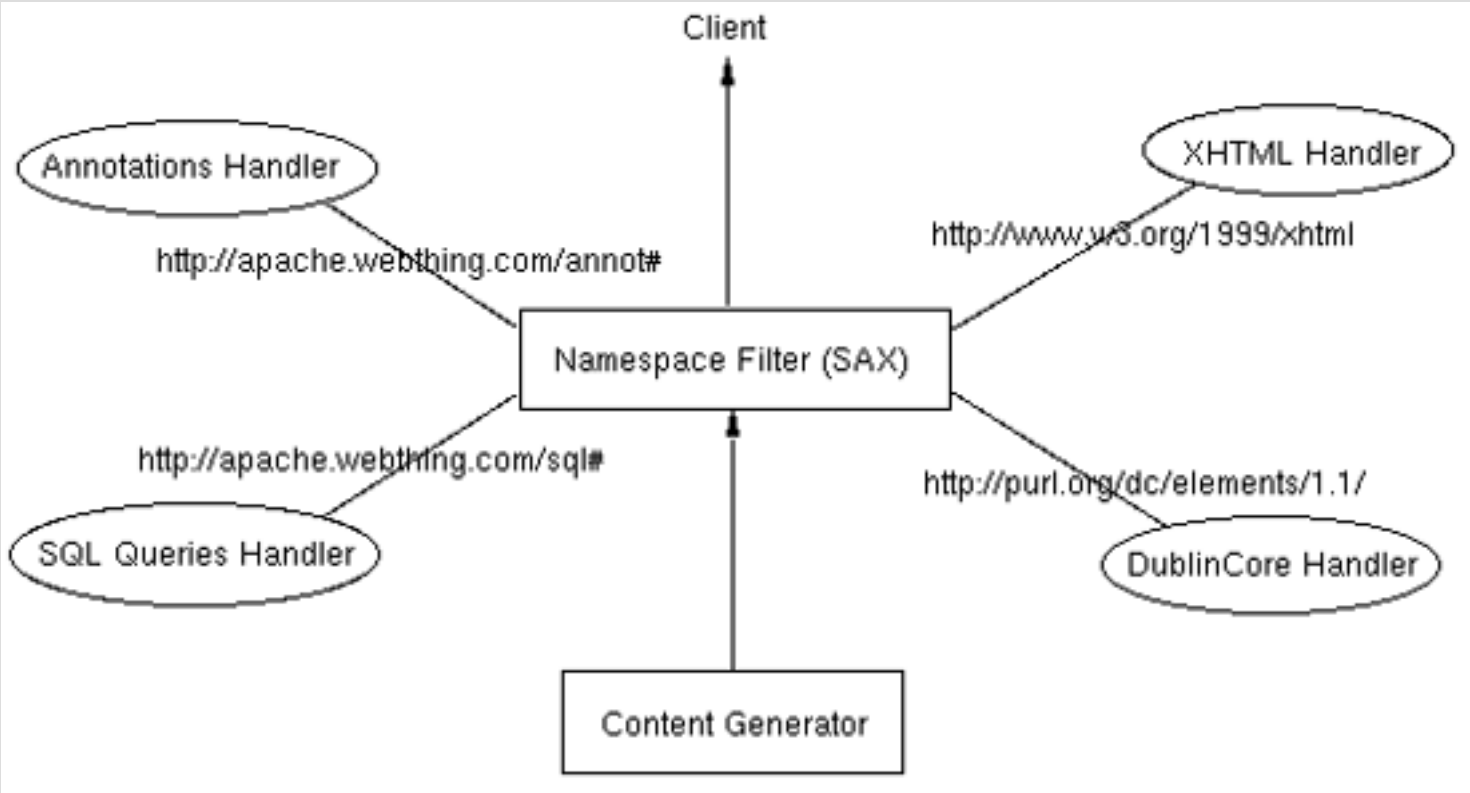
http://apache.webthing.com/mod_accessibility/betsie.html

- SAX parser -> speed and scalability
- any Content Generator

Reverse Proxy (2003)



XML namespaces (2003)



Accelerator Proxy (2004)

- Mobile Phones
- Low bandwidth, high latency
- Client capabilities string

- Goal: Aggressively minimise byte count for all media types.
- Client keepalive competes with parallelism

Reducing Byte Count

- Compress all compressible types (eg text)
- HTML: strip everything non-significant based on DTD representing the browser
- Images: Downsample to fit screen size & palette

Many filters in a single proxy - problems

- Dispatching to the right filters
- Coping with malformed input

Issues Arising

Many filters in a single proxy – problems

- Dispatching to the right filters --> mod_filter
- Coping with malformed input --> correction

Non-problems

- SSL and Compressed input work just fine within the existing filter framework (though an additional decompression filter was required in mod_deflate)

Text Transformation Steps

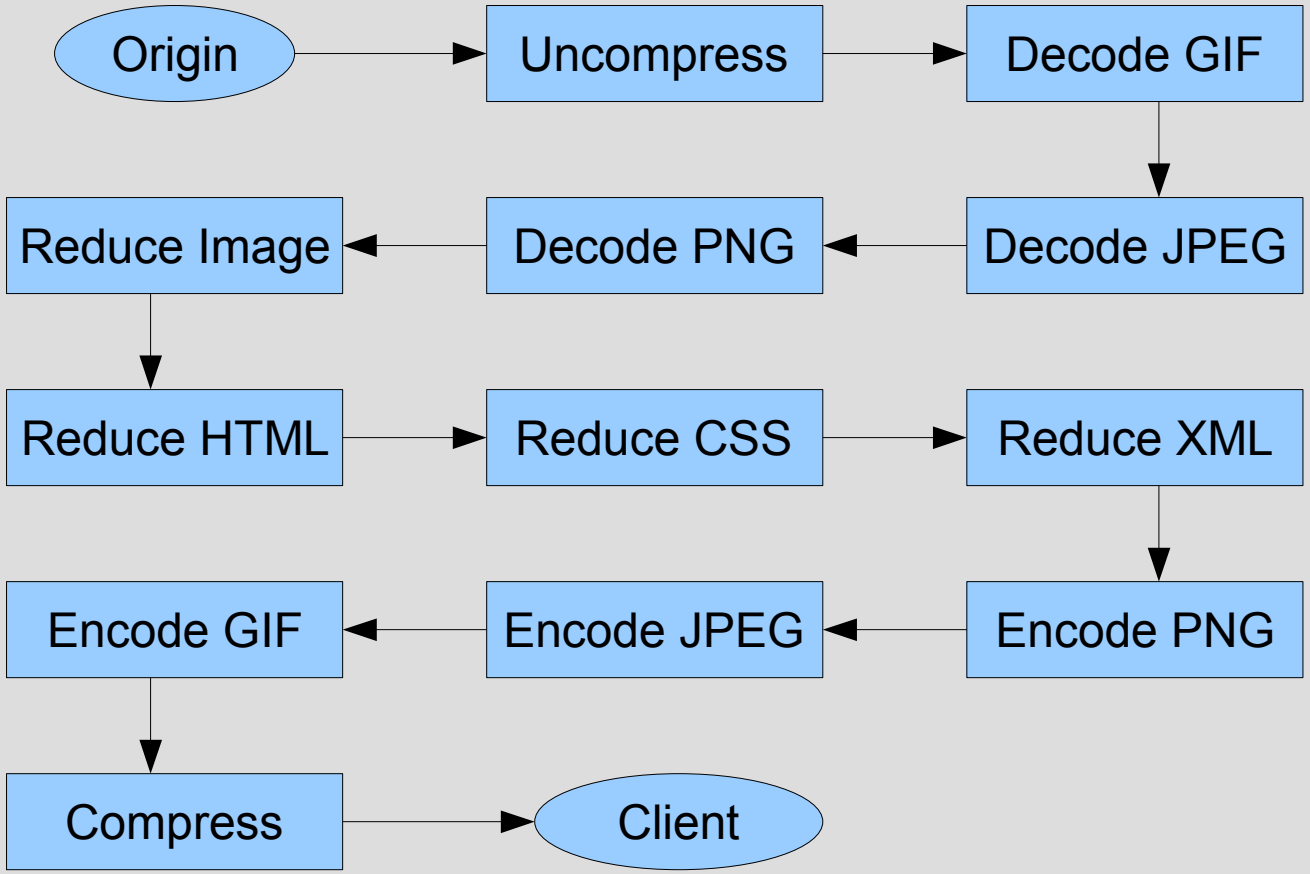
- (Decrypt)
- (Uncompress)
- (Transcode)
- (Error Correction)
- Application Filter
- (Transcode)
- (Compress)
- (Encrypt)

Image Transformation Steps

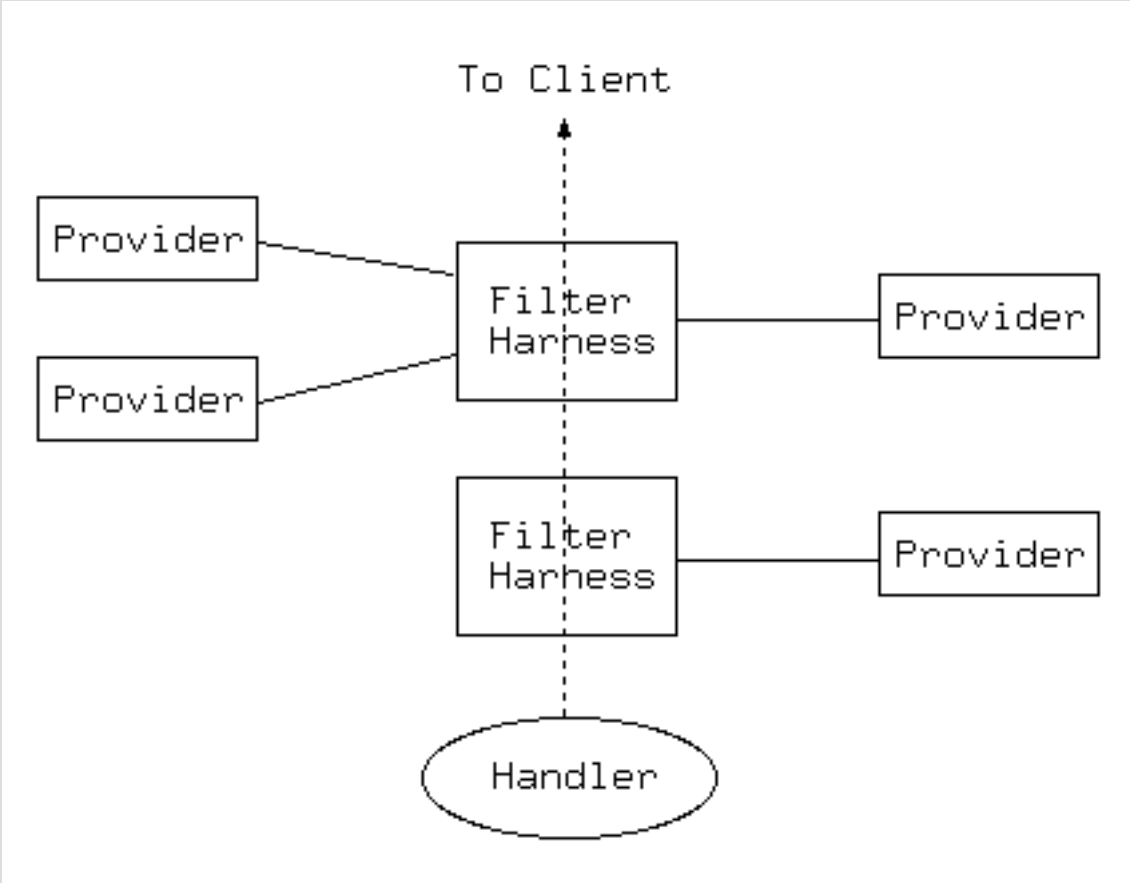
- (Decrypt)
- Unencode [gif/jpeg/png] --> pixels
- Image Processing Filters
- Encode pixels to [gif/jpeg/png]
- (Encrypt)

Note: this works for arbitrary input types, including multi-layer, multi-source input types such as encountered in GIS or CAD.

Multi-type filter



Dispatching: mod_filter

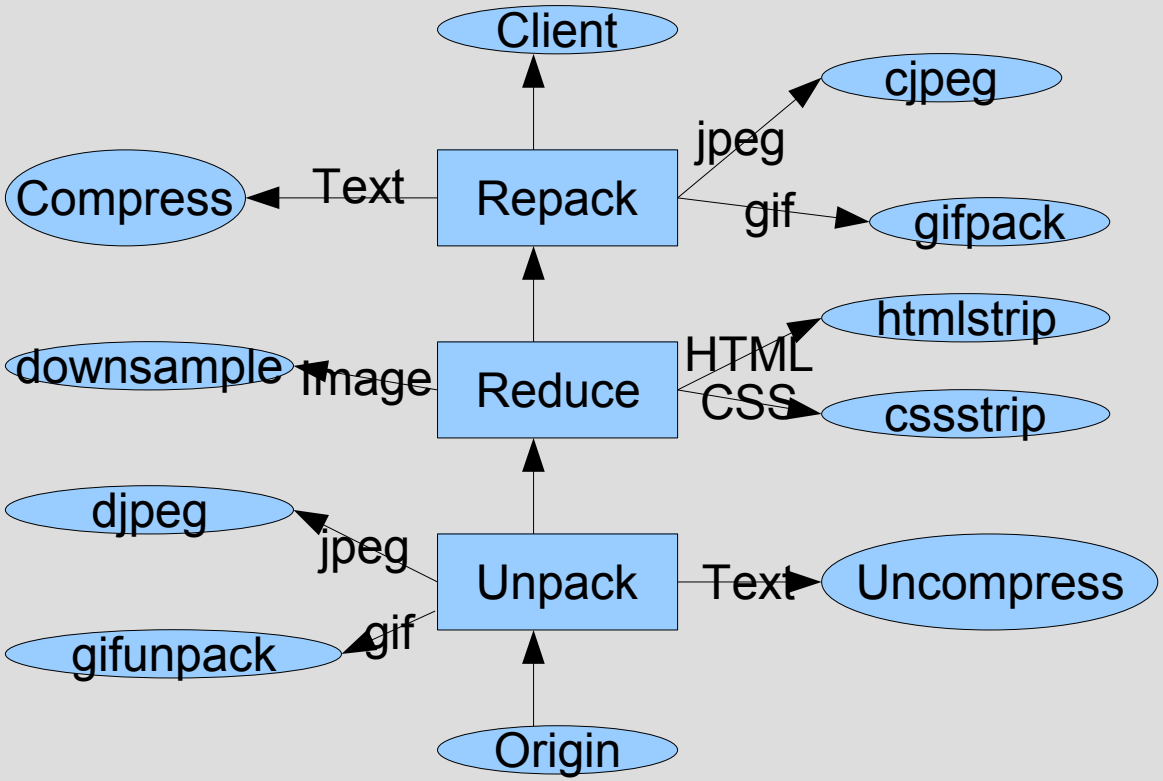


Multi-type filter

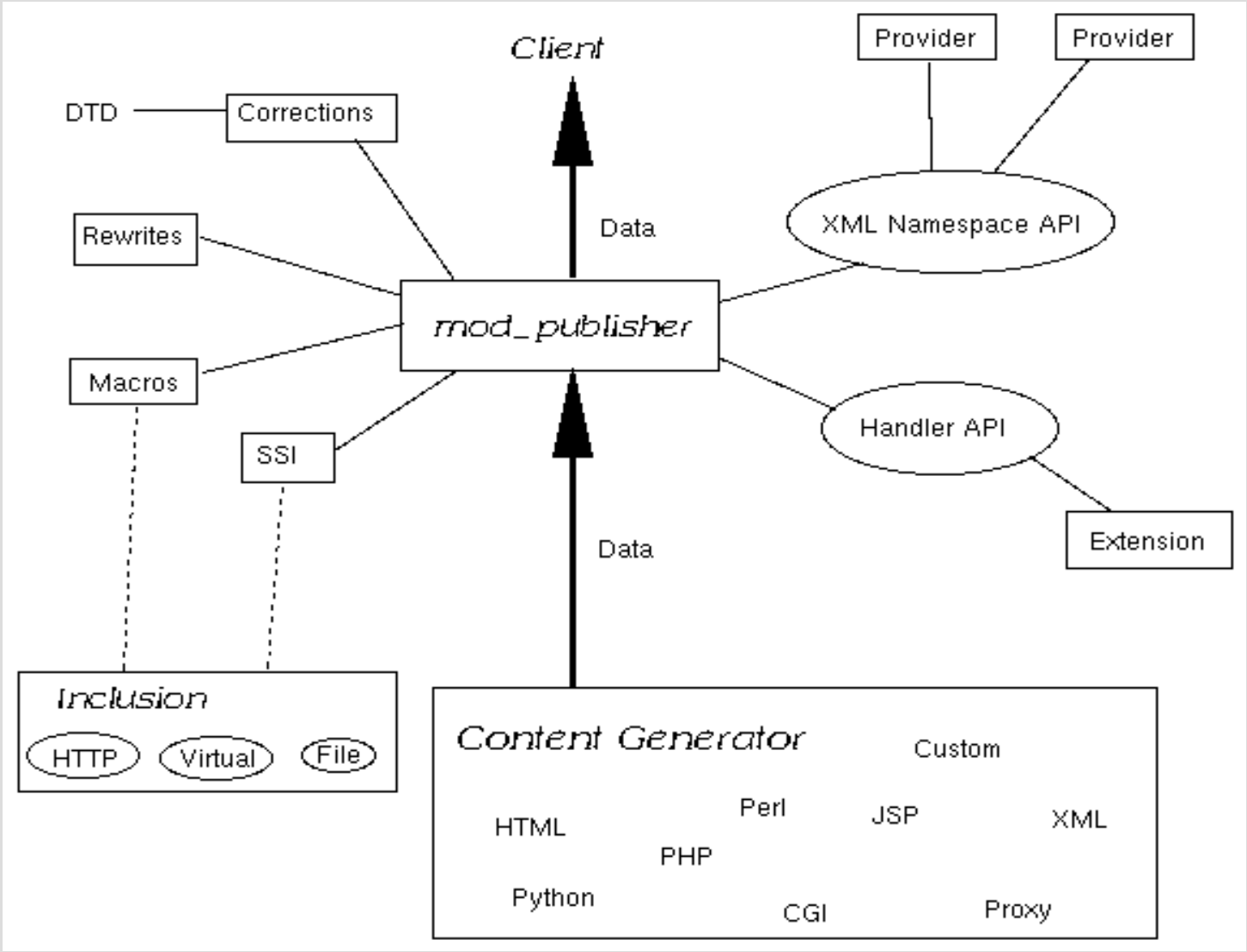
```
FilterProvider unpack INFLATE $text
FilterProvider unpack djpeg image/jpeg
FilterProvider unpack gifunpack image/gif
FilterProvider reduce htmlstrip $text/html
FilterProvider reduce downsample $image
FilterProvider repack DEFLATE $text
FilterProvider repack cjpeg image/jpeg
FilterProvider repack gifpack image/gif

FilterChain unpack reduce repack
```

Multi-type filter



Generic Markup Filtering (2004)



Generic Text Filtering

- mod_line_edit: “sed”-like editor
- Line-oriented: overcomes the problem of arbitrary breaks whilst streaming
- Generic editing for non-markup text
- Can also be used in front of a markup filter to correct input errors

Example - Markup Fixup

- Normalisation to HTML 4 or XHTML 1

```
<p align=”right”>paragraph text
<font ...>different text</font>
and some more text.</p>
```

- Text-only representation

```
<img src=”apachecon.png” alt=”ApacheCon”>
```

Example – Analysis: TOC

<h1> ... </h1>
bla bla bla
<h2> ... </h2>
.....
<table summary=”....” otherattributes>
<caption>...</caption>
.....
</table>

Example - Linearisation

```
<table><tr>
<td>first text presented in left column</td>
<td><img src=... alt=”image in middle”></td>
<td>second text presented in right column</td>
</tr></table>

<div class=”table”>
<p>first text ...</p>
<p><img src=... alt=”image in middle”></p>
<p>second text ...</p>
</div>
```

Example – Data Discovery

```
<p>To learn more, <a href=”more.html”>click
   here</a>.</p>

<p>To learn more, <a href=”more.html”
   title=”Data Discovery”>click here</a>.</p>
```

Example - Templating

```
var sponsor file /path/to/sponsors.inc
```

startElement handler:

```
<sponsor/>
<p class="sponsor">Sponsored by ...</p>
```

Comment Handler

```
<!--#include file="/path/to/sponsors.inc"-->
<p class="sponsor">Sponsored by ...</p>
```

Example – Sitewide markup

<head>, </head>, <body>, </body> are implied,
so we can hook site-wide templates on them.

```
<link rel="stylesheet" type="text/css" href="style.css"
   />
</head>
<body>
<div id="logo"><a href="/"><img src="logo.gif" alt="My
   Site"/></a></div>
body text ...
<div id="navbar"> ... </div>
<div id="footer"> ... </div>
</body>
```

References

Applications Development with Apache
(the apache modules book).
http://httpd.apache.org/docs/2.2/mod/mod_filter.html,
mod_proxy.html, mod_cache.html
http://apache.webthing.com/ (best selection of filters for text and markup).
http://apache.webthing.com/xmlns.html
http://www.apachetutor.org/apps/annot
http://www.apacheweek.com/features/reverseproxies

http://www.miswebdesign.com/resources/articles/mod_accessibility.html
http://www.xml.com/pub/a/2004/12/15/apache-namespaces.html
http://www.idealliance.org/xtech/05/call/xmlpapers/02-04-02.1566/.02-04-02.html