Web Development using Linux

“”

Contents


```
 This book seeks to provide
  ⇒ recipes for developing web
  ⇒ sites
 using the Linux operating system,
  ⇒  with an emphasis on command
  ⇒ line
 tools. This book is not about
  ⇒ html and css since that is
 covered in the html-css-book.txt

IMAGES

IMAGE COMPRESSION ....

  == tools for image compression
  .. webpack -
  .. pngout -
  ..

IMAGE RESIZING  ....

  * Convert a bunch of HTML files
    ⇒ from ISO-8859-1 to UTF-8
    ⇒ file
     >> for x in `find . -name '*.
      ⇒ html'` ; do iconv -f ISO
      ⇒ -8859-1 -t UTF-8 $x > "$x.
      ⇒ utf8"; rm $x; mv "$x.utf8"
      ⇒  $x; done

FILE TRANSFER

  rsync, ftp, sftp

WEB SITE MIRRORING

  * Download all images from a
    ⇒ site
  >> wget -r -l1 --no-parent -nH -
    ⇒ nd -P/tmp -A".gif,.jpg" http
    ⇒ ://example.com/images
```

```
POSTING DATA

  * Submit data to a HTML form
    ⇒ with POST method and save
    ⇒ the response
  >> curl -sd 'rid=value&submit=
    ⇒ SUBMIT' <URL> > out.html

  * post with a proxy and
    ⇒ authentication
  >> curl -F name='../htdocs/notes
    ⇒ /'$1 -F contents='<'$1 -u
    ⇒ user:upass -x prox.net:8080
    ⇒ -U bob:proxpass http://serv.
    ⇒ net/save.cgi
HTML STUFF

HTML LINKS ....

  * get the links from a page
  >> lynx -dump -listonly www.
    ⇒ server.net/page.html

  * find urls within an html file
    ⇒ (most of them anyway)
  >> egrep 'https?://([[:alpha
    ⇒ :]]([-[:alnum:]]+[[:alnum
    ⇒ :]])*\.)+[[:alpha:]]{2,3}(:
    ⇒ \d+)?(/([-\w/_\.]*(\?\S+)?)
    ⇒ ?)?'

ENTITIES ....

  * encode HTML entities
  >> perl -MHTML::Entities -ne '
    ⇒ print encode_entities($_)' /
    ⇒ tmp/subor.txt

  * or use xmlstarlet to encode
    ⇒ entities.

BASH AND WEB DEVELOPMENT
```

```
  Using the bash shell to develope
   ⇒  websites maybe quite
  efficient, if unconventional.

  * possibly the simplest way to
   ⇒ create a webpage from text
  >> cat page.txt | (echo '<html><
   ⇒ body><pre>'; cat -; echo '</
   ⇒ pre></body></html>')

TEMPLATING WITH BASH ....


  * a simple template technique
   ⇒ with bash
  >> export a=b; echo -e 'one\ntwo
   ⇒ \nand <x>' | (echo 'cat <<
   ⇒ EE';sed 's/<x>/$a/g'; echo '
   ⇒ EE') | bash

  * use the technique above to
   ⇒ substitute the date into the
   ⇒  template
  >> cat template | (echo 'cat <<
   ⇒ EE';sed 's/<date>/$(date)/g
   ⇒ '; echo 'EE') | bash

FOLDER LISTINGS ....

  * list only folders
  >> ls -d */ | (echo '<ul class="
   ⇒ fol">'; cat -; echo '</ul>')

  * make an html directory listing
   ⇒  out of the current folder
  >> echo "echo -e \<li\>"{$(echo
   ⇒ * | tr ' ' ',')}"\</li\>" |
   ⇒ bash

  * list all files and folders, no
   ⇒  links
  >> a=$(echo *); echo 'echo -e "\
   ⇒ n<li>"{'${a//" "/,}'}"</li
   ⇒ >"' | bash

  * list only folders, no links
  >> a=$(echo */); echo 'echo -e
   ⇒ "\n<li>"{'${a//" "/,}'}"</li
   ⇒ >"' | bash

  * a for loop method to list only
   ⇒  sub-folders as an html list
  \begin{lstlisting}
    echo "<ul class=fol>"
    for d in $(ls -d */); do
      echo "<li>$d</li>"
    done
    echo "</ul>"
```

* another for loop method to list only sub-folders as an html list

```
    echo "<ul class=fol>"
    for d in */; do
      echo "<li>$d</li>"
    done
    echo "</ul>"
```

* list subfolders as html links

```
    echo "<ul class=fol>"
    for d in $(ls -d */); do
      echo "<li><a href='$d'>$d</a
       ⇒ ></li>"
    done
    echo "</ul>"
```

XHTML
== tools .. xmlstarlet - queries and edits xml from the command line ..
CURL STUFF
* getting a page via an authenticating proxy server ¿¿ curl -x proxy.utas.edu.au:8080 -U bobj http://www.server.net
* getting a page via an authenticating proxy server as user 'bobj' with password 'asecret' ¿¿ curl -x proxy.net:8080 -U bobj:asecret http://www.server.net
Supplying the password in this manner is possibly not a good idea from a security point of view
* download a text file through a proxy and edit it with vim

```
  function edn
  {
    curl -x proxy.org.au:8080 -U
     ⇒ bob:pass www.serv.net/a.
     ⇒ txt -o ~/notes.txt
    vim ~/notes.txt
  }
```

* upload a file to webserver w cgi script with http authententication,

```
function up {
 [ -z "$1" ] && echo 'no
  ⇒ parameter' && return 1;
 curl -F name='../htdocs/notes/'
  ⇒ $1 -F contents='<'$1 -u user
  ⇒ :upass -x prox.net:8080 -U
  ⇒ bob:proxpass http://serv.net
  ⇒ /save.cgi
}
```

PERL TRICKS
== useful modules .. www::mechanize .. lwp ..
* a perl mechanize example

```
# navigate to the main page
$mech->get('http://www.
 ⇒ somesite.com/');

# follow a link that contains
 ⇒ the text 'download this'
$mech->follow_link( text_regex
 ⇒  => qr/download this/i );

# submit a POST form, to log
 ⇒ into the site
$mech->submit_form(
with_fields      => {
username    => 'mungo',
password    => 'lost-and-alone
 ⇒ ',
}
);

# save the results as a file
$mech->save_content('somefile.
 ⇒ zip');
```

PHP
* Testing php configuration ¿¿ php -r "phpinfo;"
* get the urls from a webpage ¿¿ $browser->getUrls()$
CRAWLING
.. ScriptableBrowser (simpletest)
LOAD TESTING
== tools .. funkload - web testing ..
STUFF USED IN GOOGLE CHROME
bsdiff, bspatch, bzip2, dtoa, hunspell, ICU, JSCRE, libjpeg, libpng, libxml, libxslt, LZMA SDK, modp$_b$64, $MozillainterfacetoJavaPluginAPIs$, npapi, nspr, nss, sfntly, skia, sqlite, tcmalloc, v8 assembler, WebKit, zlib
FORUM SITES
@@ ask.metafilter.com visited by knowledgable people @@
SERVING PAGES
* share the current directory tree (via http) at http://$HOSTNAME$: 8000/ >> $python -mSimpleHTTPServer$
* create a webserver to share all files in /tmp/mydocs on port 8081 ¿¿ wbox servermode webroot /tmp/mydocs
* create a webserver to share all files in /tmp/mydocs on port 8080 ¿¿ wbox servermode serverport 8080 webroot /tmp/mydocs
* Sharing file through http 80 port ¿¿ nc -w 5 -v -l -p 80 ¡ file.ext
CGI SERVERS
* a simple cgi server ¿¿ python -m CGIHTTPServer 8080
* some kind of perl cgi server ¿¿ HTTP::Simple::PSGI
SMALL WEBSERVERS
@@ http://ask.metafilter.com/65481/Help-me-find-a-cool-little-unix-http-utility-I-cant-remember some good things about mini web servers @@ http://hping.org/wbox/ site for wbox
== small quick and easy webservers .. wbox - http server .. thttpd - small web server .. mini$_h$$ttpd - same author as thttpd but smaller .. webfs - serves a file system from the web .. busybox httpd command - small web server everything ..$
WBOX
developed from 2007 - 2009 ...
* show how long each part of a webpage takes to generate ¿¿ wbox nowhere.net/page.html timesplit 1
* show the http header information for a page ¿¿ wbox www.google.it/notexistingpage.html 1 showhdr
MAC OSX
Installed by default are php, curl,
TEMPLATING
@@ http://www.perl.com/pub/a/2001/08/21/templating an article about using templating systems with perl
== tools .. Template Toolkit - almost active development, perl, and python .. HTML::Mason - ? callback style, active as of 2010 .. Embperl - embedd perl into webpages .. stopped 2006 .. HTML::Template - 8 perl template module .. Text::Template - a general purpose templater .. Apache::ASP - use asp with apache, stopped 2004 .. CGI::FastTemplate - another one ..
@@ http://template-toolkit.org/ the site for the template toolkit
TEMPLATE TOOLKIT
* example statement using dot notation. ¿¿ How are things in [
* a for loop [¡a href="[[

Section 1

Html Template

Example loop with html::template —— <TMPL_LOOP list> <a href="<TMPL_VAR url>"><TMPL_VAR name></A> </TMPL_LOOP> ,,,

1.1 Mason

Seems to be in active development. It can be run without a webserver

`http://www.masonhq.com/`
: the official site

Install mason using apt-get

```
apt-get install libmason-perl   ~(??? unchecked)
```

Perl Stuff

some interesting perl web modules

`Dancer`	Perl web apps with good examples
`Web::Simple`	Simple web apps
`CGI::Application`	
`Catalyst`	Big web apps
`Mason`	
`Mojolicious`	

2.1 Web Simple

`http://search.cpan.org/~mstrout/Web-Simple-0.002/lib/Web/Simple.pm`
: the documentation for web simple

Developed in 2009. Can create a webapplication without a webserver

`hobbs`
: at stackoverflow.com knowledgable perl web person

Cpan Tool Crash Course

Start a cpan shell to install mason

```
perl -MCPAN -e 'shell'
```

simple cpan shell

`h`	Show help
`get`	Get the source for a module
`make`	Compile ? the module
`test`	Test a module
`install`	Do all of get, make, test
`clean`	Get rid of a badly installed module
`look`	See whats happening
`readme`	See whats going on

`http://www.livejournal.com/doc/server/lj.install.perl_setup.modules.html`
: a list of potentially useful modules used with livejournal

Upgrade cpan but could cause problems ???

```
perl -MCPAN -e shell
cpan> install Bundle::CPAN
cpan> reload cpan
```


Getting Web Pages From The Command Line

Http get of a web page via proxy server with login credentials

```
curl -U username[:password] -x proxy:proxyport webpage
```

Use netcat to get a webpage

```
echo "GET / HTTP/1.0\r\r" | nc -v www.somewebsite.com 80
```

Get a webpage with the console php tool

```
php -r "file('http://metafilter.com/');"
```

Get a webpage with perl

```
perl -MHTTP::Client -e 'print HTTP::Client->new()->get("http://
  ⇒ localhost/path")'
```

A simpler way to do the same

```
perl -MLWP::Simple -e 'get("http://www.metafilter.com/")'
```

Section 5

Load Testing

Perform 4 queries per second (with 4 processes) on the local webserver

```
wbox http://localhost clients 4
```

Section 6

Blog Engines

`serendipity` A blog engine with database backend

Section 7

Notes

google use python

`http://unixmages.com/`
: an unrelated but interesting site