Gnome's Guide to WEBrick

Yohanes Santoso ¡ggw¡at¿microjet.ath.cx¿

Version 0.6, 2004.09.19

Abstract

So, you installed Ruby 1.8. So, you wanted to do some web development. So, you heard about this thing called WEBrick that comes standard with Ruby. So, you googled for documentation. So, all you could find was Eric Hodel's articles. So, you thought, "Gosh, where is the documentation?". So, you were feeling brave and tried to "use the source, Luke". So, you realised that you were a newbie to Ruby and the source looked like Yoda doing Yoga while chanting "may the force be with you". So, you didn't think the force would be with you for at least another week. So, you were feeling impatient because you want to start trying now and perhaps you could have practised the force while trying. So, you finally screamed, "I've had it!".

So, I hope you will find this adequate to start and accompany you during your journey into WEBrick. There is also the reference section which you can refer to when you are lost or unsure, or both.

Note: this documentation references the WEBrick code shipped with Ruby 1.8.1.

Contents

1 What is WEBrick

WEBrick is a HTTP server library written by TAKAHASHI Masayoshi, GOTOU Yuuzou, along with patches contributed various Ruby users and developers. It started from an article entitled "Internet Programming with Ruby" in a Japanese network engineering magazine "OpenDesign". And now, it is part of the Ruby 1.8 standard library.

You can use WEBrick to create HTTP-based server or application. You may use also use it as a base for building web-application frameworks like IOWA, Tofu, and many others.

It can also be used to build non-HTTP server, like the Daytime Server example in the WEBrick home page, although that would be a pity since you would not be able to use WEBrick's support for the HTTP protocol.

In the web-application paradigm, WEBrick is quite low-level. It does not know about "web application", for starter. "user interaction session" is also a foreign concept.

All it knows are servlets. As far as it concerns, each servlet is independent from the others. If there are many servlets working together to provide a web application, guess who should provide the glue? You! If you want to track a user's interaction through the servlets, guess who should provide the code? You! If you need those functionalities, I recommend using IOWA or Tofu or others. Other people have took pain to provide that additional layers on top of WEBrick, so you do not have to re-invent the wheel.

```
# A simple WEBrick invocation

require 'webrick'

server = WEBrick::HTTPServer.new

#
# You would want to mount handlers here. Read further to know what
# handlers are.
#

# trap signals to invoke the shutdown procedure cleanly
['INT', 'TERM'].each {|signal| trap(signal) {server.shutdown}}

server.start
```

The above example will start WEBrick with the default configuration, including the configuration that tells it to listen on port 80. Now, let us try to override some of the configuration:

1. Listen on port 8080 instead of port 80 (for the rest of this documentation, the default listening port is port 8080 since I already reserve port 80 for the Apache HTTP Server that always runs on my machine.
2. Serve files from the directory `/var/www`

To do the above, one would need to pass the appropriate configuration when instantiating the `HTTPServer`. Since for the rest of this document we are going to modify the configuration and instantiate `HTTPServer` quite frequently, let us also ease that process by defining a method that does all that.

```
require 'webrick'

include WEBrick    # let's import the namespace so I don't have to
                   # keep typing WEBrick:: in this documentation.
```

```
def start_webrick(config = {})

  config.update(:Port => 8080)     # always listen on port 8080
  server = HTTPServer.new(config)
  yield server if block_given?
  ['INT', 'TERM'].each {|signal| trap(signal) {server.shutdown}}
  server.start

end

start_webrick(:DocumentRoot => '/var/www')
```

Output:

```
dede:~$ w3m -dump http://localhost:8080
Index of /

 Name                           Last modified         Size
------------------------------------------------------------------------------

 Parent Directory               2004/07/18 06:51      -
 docbook-dsssl/                 2003/10/15 00:30      -
 pub/                           2004/05/24 15:46      -

------------------------------------------------------------------------------

WEBrick/1.3.1 (Ruby/1.8.1/2004-02-03)
at localhost:8080
```

2 Mounting Servlets

In WEBrick terminology, mounting means setting up an instance of a subclass of `HTTPServlet::AbstractServlet` a.k.a "servlet" to service a request-URI.

When mounting a servlet, one should specify the prefix of the request-URI it services. If there are more than one mounts that match the request-URI, the one with the closest match is selected. For example: a servlet mounted at `/foo` would probably service the request-URI `/foo/bar/is/foolish`, but if there is another servlet mounted at `/foo/bar`, then that servlet would be the one selected instead.

To mount the servlet, specify the mount the path along with the class of the servlet. WEBrick creates a new instance from the servlet class for each request it receives, and executes it in a separate thread.

```
class FooServlet < HTTPServlet::AbstractServlet
end
class FooBarServlet < HTTPServlet::AbstractServlet
```

```
end

start_webrick {|server|
  server.mount('/foo', FooServlet)
  server.mount('/foo/bar', FooBarServlet)
}
```

3 Standard Servlets

WEBrick comes with several servlets that you can use right away.

- `HTTPServlet::FileHandler`
- `HTTPServlet::ProcHandler`
- `HTTPServlet::CGIHandler`
- `HTTPServlet::ERBHandler`

3.1 FileHandler

`FileHandler` is one of the more useful standard servlets. If you specified the `:DocumentRoot` option, WEBrick will install a `FileHandler` configured to serve the path specified in the option. Roughly, WEBrick automate the following for you when you set the `:DocumentRoot` option.

The example shown in an earlier section,

```
start_webrick(:DocumentRoot => '/var/www')
```

is functionally similar to

```
start_webrick {|server|
  doc_root = '/var/www'
  server.mount("/", HTTPServlet::FileHandler, doc_root,
  {:FancyIndexing=>true})
}
```

The above example pass the `:FancyIndexing` option to the `FileHandler` servlet. There are more options described in the FileHandler Configuration section.

If the request path refers to a directory, `FileHandler` serves the directory index file. If there is no directory index file, and the `:FancyIndexing` option is specified, it will serves the directory listing, otherwise it will return a 403 status (Forbidden)

3.1.1 Overriding Default MIME Type

`FileHandler` needs to know the mime-type of the file so it can set the `Content-Type` header in the HTTP response properly. For that purpose, it derives the MIME type of a file by matching the filename extension with a table of `ext => mimetype`. The default table is in `Utils::DefaultMimeTypes` and is adequate for many occasions.

However, should you find it to be inadequate, or perhaps you want to use the Apache-style mime type file in your system (usually at `/etc/mime.types`), then you can configure WEBrick to use that.

```
system_mime_table = Utils::load_mime_types('/etc/mime.types')
my_mime_table = system_mime_table.update({ "foo" => "application/foo" })

start_webrick(:MimeTypes => my_mime_table)
```

3.1.2 Default File Handler

When `FileHandler` receives a request, it analyse the request path. It will delegates the request handling to `DefaultFileHandler` if the request path:

- does not end with `.cgi`, or
- does not end with `.rhtml`

`DefaultFileHandler` emits a `ETag` header based on the file's inode (what is the inode value in non-Unix, nil?), size and modification time. It understands some other the request headers. Below is a list the HTTP headers it services along with the corresponding explanation from the HTTP/1.1 RFC.

if-modified-since The If-Modified-Since request-header field is used with a method to make it conditional: if the requested variant has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body.

if-none-match The If-None-Match request-header field is used with a method to make it conditional. A client that has one or more entities previously obtained from the resource can verify that none of those entities is current by including a list of their associated entity tags in the If-None-Match header field. The purpose of this feature is to allow efficient updates of cached information with a minimum amount of transaction overhead. It is also used to prevent a method (e.g. PUT) from inadvertently modifying an existing resource when the client believes that the resource does not exist.

if-range If a client has a partial copy of an entity in its cache, and wishes to have an up-to-date copy of the entire entity in its cache, it could use the

Range request-header with a conditional GET (using either or both of If-Unmodified-Since and If-Match.) However, if the condition fails because the entity has been modified, the client would then have to make a second request to obtain the entire current entity-body.

The If-Range header allows a client to "short-circuit" the second request. Informally, its meaning is 'if the entity is unchanged, send me the part(s) that I am missing; otherwise, send me the entire new entity'.

range The presence of a Range header in an unconditional GET modifies what is returned if the GET is otherwise successful. In other words, the response carries a status code of 206 (Partial Content) instead of 200 (OK).

The presence of a Range header in a conditional GET (a request using one or both of If-Modified-Since and If-None-Match, or one or both of If-Unmodified-Since and If-Match) modifies what is returned if the GET is otherwise successful and the condition is true. It does not affect the 304 (Not Modified) response returned if the conditional is false.

3.2 CGIHandler

What if you have some CGI programs that you do not want or do not have the time to rewrite as WEBrick servlets? Worry not for you can still use them. Simply install a `FileHandler` on the directory containing your CGIs and make sure that the program files have a `.cgi` suffix.

```
start_webrick {|server|
  cgi_dir = File.expand_path('~ysantoso/public_html/cgi-bin')
  server.mount("/cgi-bin", HTTPServlet::FileHandler, cgi_dir, {:FancyIndexing=>true})
}

dede:~$ cat ~ysantoso/public_html/cgi-bin/test.cgi
#!/usr/bin/env ruby
print "Content-type: text/plain\r\n\r\n"
ENV.keys.sort.each{|k| puts "#{k} ==> #{ENV[k]}"}

dede:~$ w3m -dump http://localhost:8080/cgi-bin/test.cgi
GATEWAY_INTERFACE ==> CGI/1.1
HTTP_ACCEPT ==> text/*, image/*, application/*, video/*, audio/*, message/*
HTTP_ACCEPT_ENCODING ==> gzip, compress, bzip, bzip2, deflate
HTTP_ACCEPT_LANGUAGE ==> en;q=1.0
HTTP_HOST ==> localhost:8080
HTTP_USER_AGENT ==> w3m/0.5.1
PATH_INFO ==>
QUERY_STRING ==>
REMOTE_ADDR ==> 127.0.0.1
REMOTE_HOST ==> dede
REQUEST_METHOD ==> GET
```

```
REQUEST_URI ==> http://localhost:8080/cgi-bin/test.cgi
SCRIPT_FILENAME ==> /home/ysantoso/public_html/cgi-bin/test.cgi
SCRIPT_NAME ==> /cgi-bin/test.cgi
SERVER_NAME ==> localhost
SERVER_PORT ==> 8080
SERVER_PROTOCOL ==> HTTP/1.1
SERVER_SOFTWARE ==> WEBrick/1.3.1 (Ruby/1.8.1/2004-02-03)
```

When `FileHandler` sees that the request path ends with `.cgi`, it delegates the request to `CGIHandler`. Then, `CGIHandler` setup the necessary CGI-related environment variables and run the requested CGI program. The CGI program can affect the HTTP response status returned by WEBrick by setting the header "status" to the desired response number.

```
dede:~$ cat ~ysantoso/public_html/cgi-bin/test.410.cgi
#!/usr/bin/ruby
print "Status: 410"
print "Content-type: text/plain\r\n\r\n"
puts "Tired. Frustrated. Too many requests. Gone fishing. Be back after 5pm."

dede:~$ w3m -dump_extra http://localhost:8080/cgi-bin/test.410.cgi
W3m-current-url: http://localhost:8080/cgi-bin/test.410.cgi
W3m-document-charset: US-ASCII
HTTP/1.1 410 Gone
Connection: close
Date: Sun, 19 Sep 2004 22:33:25 GMT
Server: WEBrick/1.3.1 (Ruby/1.8.1/2004-02-03)
Content-Length: 71

Tired. Frustrated. Too many requests. Gone fishing. Be back after 5pm.
```

Warning: `CGIHandler` waits until the called CGI process finishes. If your CGI performs incremental output, the output will not be sent back to client until after the CGI process exits. I have been told by someone (I know the name but I do not want to mention it because I do not want to push him to commit to this) that he will try to get another CGI handler, that sends back the output immediately, for inclusion in Ruby 1.8.2.

3.3 ERBHandler !NOT YET.

I have no idea what ERB is.

3.4 ProcHandler

WEBrick allows you to be lazy. If your need is trivial and can be expressed in a simple Proc or a block, then you don't have to bother with subclassing `AbstractServlet`.

```
start_webrick {|server|
  server.mount_proc('/myblock') {|req, resp| resp.body = 'a block mounted at #{req.script_na

  my_wonderful_proc = Proc.new {|req, resp| resp.body = 'my wonderful proc mounted at #{req.
  server.mount_proc('/myproc', my_wonderful_proc)

  server.mount('/myprochandler', HTTPServlet::ProcHandler.new(my_wonderful_proc))
}
```

Output:

```
dede:~$ w3m -dump http://localhost:8080/myblock
a block mounted at /myblock
dede:~$ w3m -dump http://localhost:8080/myproc
my wonderful proc mounted at /myproc
dede:~$ w3m -dump http://localhost:8080/myprochandler
my wonderful proc mounted at /myprochandler
```

4 Writing a Custom Servlet

4.1 The do_ Methods

Writing a servlet is easy enough. First, you need to create a subclass of `HTTPServlet::AbstractServlet`. Then, depending on whether you want to service GET or POST or OPTIONS or HEAD request, you add a `do_GET` or `do_POST` or `do_OPTIONS` or `do_HEAD` method respectively. If you want to support some of the less-frequently-encountered request, like PUT, you just need to create a corresponding `do_` method, e.g.: `do_PUT`.

`AbstractServlet` implements a `do_HEAD` and `do_OPTIONS` for you. `do_HEAD` simply calls `do_GET` (which you need to provide) and sends back everything except the body. `do_OPTIONS` simply return a list of `do_` methods available.

"What should a `do_` method do?", you asked. That is up to you. WEBrick will call your `do_` method with two arguments: the request and the response objects. Normally, you'd want to, perhaps, query the request object and set the response object correspondingly.

```
class GreetingServlet < HTTPServlet::AbstractServlet
  def do_GET(req, resp)
    if req.query['name']
      resp.body = "#{@options[0]} #{req.query['name']}. #{@options[1]}"
      raise HTTPStatus::OK
    else
      raise HTTPStatus::PreconditionFailed.new("missing attribute: 'name'")
    end
  end
  alias do_POST, do_GET    # let's accept POST request too.
```

```
end

start_webrick {|server| server.mount('/greet', GreetingServlet, 'Hi', 'Are you having a nice
```

Output:

```
dede:~$ w3m -dump 'http://localhost:8080/greet'
Precondition Failed

missing attribute: 'name'
--------------------------------------------------------------------------------

WEBrick/1.3.1 (Ruby/1.8.1/2004-02-03) at localhost:8080
dede:~$ w3m -dump 'http://localhost:8080/greet?name=Gadis+Manis'
Hi Gadis Manis. Are you having a nice day?
```

4.2 Responding

There are two ways to set the response status. The first, as shown above, is to raise a `HTTPStatus` exception. I recommend this method because, in case of error status, it returns a html page filled with the backtrace. If you need to provide a custom error page,

1. Set the response status and body manually, OR
2. Extend the HTTPResponse object with the `create_error_page` method which will be called upon error.

I favour the first approach since you cannot access the exception that was thrown from within a `create_error_page` method.

```
class GreetingWithCustomisedErrorPageServlet < HTTPServlet::AbstractServlet
  def do_GET(req, resp)
    if req.query['name']
      resp.body = "#{@options[0]} #{req.query['name']}. #{@options[1]}"
      raise HTTPStatus::OK
    else
      resp.status = 412
      resp.body = "Error within GreetingWithCustomisedErrorPageServlet"
      resp['content-type'] = 'text/plain'
    end
  end
end

class GreetingWithExtendedResponseObjectServlet < HTTPServlet::AbstractServlet
  def do_GET(req, resp)
```

```
    # Extend the resp object
    class << resp
      def create_error_page
        self['content-type'] = 'text/plain'    # Default to 'text/html'
        self.body = "Error within GreetingWithExtendedResponseObjectServlet"
        # Response status is determined from the HTTPStatus exception produced
      end
    end

    raise HTTPStatus::PreconditionFailed unless req.query['name']
    resp.body = "#{@options[0]} #{req.query['name']}. #{@options[1]}"
    raise HTTPStatus::OK
  end
end

start_webrick {|server|
  server.mount('/greet1', GreetingWithCustomisedErrorPageServlet, 'Hi', 'Are you having a ni
  server.mount('/greet2', GreetingWithExtendedResponseObjectServlet, 'Hi', 'Are you having a
}
```

Output:

```
dede:~$ w3m -dump http://localhost:8080/greet1
Error within GreetingWithCustomisedErrorPageServlet
dede:~$ w3m -dump http://localhost:8080/greet2
Error within GreetingWithExtendedResponseObjectServlet
```

So, what `HTTPStatus` exceptions are available? Many; you can take a look at `httpstatus.rb`, and do the following substitution on each value in the `StatusMessage` table:

- Remove all `'-'` characters
- Remove all spaces

Example:

```
irb(main):001:0> require 'webrick'; include WEBrick::HTTPStatus
=> Object
irb(main):002:0> OK
=> WEBrick::HTTPStatus::OK
irb(main):003:0> RequestURITooLarge
=> WEBrick::HTTPStatus::RequestURITooLarge
```

The body of a response does not necessarily have to be a String. You can pass an `IO` object too. This should be handy if the response is long, e.g.: returning the content of a file 16MB large.

4.3 Controlling Servlet Instantiations

Sometimes, you do not want WEBrick to automatically create a new instance of your servlet class. For example, if the initialisation part of your servlet is expensive, you may want to reuse the same instance or at least manage a pool of instances.

WEBrick calls your the class method `get_instance` with the parameters `config` and —options—. This method should return the instance that WEBrick should use to service the request. I recommend placing a mutex around critical area since now the same instance may be accessed from more than one threads simultaneously.

```
require 'thread'
class CounterServlet < HTTPServlet::AbstractServlet

  @@instance = nil
  @@instance_creation_mutex = Mutex.new
  def self.get_instance(config, *options)
    @@instance_creation_mutex.synchronize {
      @@instance = @@instance || self.new(config, *options)
    }
  end


  attr_reader :count
  attr :count_mutex
  def initialize(config, starting_count)
    super
    @count = starting_count
    @count_mutex = Mutex.new
  end

  def do_GET(req, resp)
    resp['content-type'] = 'text/plain'
    @count_mutex.synchronize {
      resp.body = @count
      @count += 1
    }
  end
end

start_webrick {|server|
  server.mount('/count_from_0', CounterServlet, 0)
  server.mount('/count_from_0_too', CounterServlet, 100) # 100 has no effect
}
```

Output:

```
dede:~$ w3m -dump http://localhost:8080/count_from_0
0
dede:~$ w3m -dump http://localhost:8080/count_from_0
1
dede:~$ w3m -dump http://localhost:8080/count_from_0
2
dede:~$ w3m -dump http://localhost:8080/count_from_0
3
dede:~$ w3m -dump http://localhost:8080/count_from_0_too
4
dede:~$ w3m -dump http://localhost:8080/count_from_0_too
5
```

4.4 Cookies

Eric Hodel has graciously allowed me to reproduce his article on WEBrick's cookies here for the benefit of hard-copy readers. The WEBrick::Cookies structure is also copied in the reference section.

4.4.1 Eric Hodel's "WEBrick and Cookies"

Source: WEBrick and Cookies

WEBrick exposes cookies in a simple, easy to use Cookie class that exposes all the properties of RFC 2109 cookies. Both the `HTTPRequest` and `HTTPResponse` handily allow you to read and set cookies on requests.

(Cookies are delicious delicacies.)

WEBrick::Cookie

`WEBrick::Cookie` is a wrapper around a cookie that exposes all the properties of a cookie. To construct a WEBrick cookie, simply call `WEBrick::Cookie.new` and provide the name and value for the cookie. After instantiating a cookie you can access cookie's properties with the following methods (descriptions from RFC 2109 and the Netscape Cookie specifications):

name The name of the cookie. The name of the cookie may only be read, not set

value The value of the cookie. value should be in a printable ASCII encoding.

version Identifies which cookie specification this cookie conforms to. 0, the default for Netscape Cookies, and 1 for RFC 2109 cookies.

domain The domain for which the cookie is valid. An explicitly specified domain must always start with a dot.

expires A Time or String representing when the cookie should expire. Expires must to be in the following format: `Wdy, DD-Mon-YYYY HH:MM:SS  GMT`

max_age The lifetime of the cookie in seconds from the time the cookie is sent. A zero value means the cookie should be discarded immediately.

comment Allows an origin server to document its intended use of a cookie. The user can inspect the information to decide whether to initiate or continue a session with this cookie.

path The subset of URLs to which the cookie applies.

secure When set to true, the cookie should only be sent back over a secure connection.

Retrieving and Setting Cookies

Cookies are read in by `WEBrick::HTTPRequest` automatically, and are available as an Array from `HTTPRequest#cookies`. When creating a `WEBrick::HTTPResponse`, cookies may be appended to the `HTTPResponse#cookies` Array.

Cookies will not be automatically copied from the `HTTPRequest` to the `HTTPResponse`. You must do this by hand.

5 Logging

WEBrick uses a logger to record its activity. This server-level logger is also made available to all servlets. Please use it to log the servlet activity instead of spewing logs after logs directly to, say, `$stderr`.

The logger has five different logging levels and a default level. Each level has its own priority and logs having a level that is of lower priority than the default level are not recorded.

The levels are (arranged from the highest to lowest priority):

- fatal
- error
- warn
- info
- debug

You may log a message by calling the logger like so: `@logger.error("1+1 is 3?  You must have been sk`
You may also want to send the `<<` message which will log the message under the info level: `@logger << "This is an info-level message"`.

The default logger has a default level of 'info' and outputs to `$stderr`, but you can change it easily enough as shown in the following example.

```
class HelloWorldServlet < HTTPServlet::AbstractServlet
  def do_GET(req, resp)
    @logger.debug("About to return 'Hello World'")
    resp.body = 'Hello World'
  end
end
```

```
# a logger that outputs to /dev/null and has a default level of 'INFO'
null_logger = Log.new('/dev/null')

# a logger that outputs to $stderr and has a default level of 'DEBUG'
fatal_stderr_logger = Log.new($stderr, Log::DEBUG)

start_webrick(:Logger => fatal_stderr_logger) {|server|
  server.mount('/helloworld', HelloWorldServlet)
}
```

5.1 Access Log

The access log is special: you are more likely to access it more frequently than the logs of other activities. As such, you may not want to do anything special to extract it from the general log. Thus, WEBrick does not mix the access log with other logs.

Well, actually the default access log and the server-level log output to the same sink: `$stderr`. Let's change it on the next example.

```
server_logger = Log.new('/var/log/webrick/server.log')


# The :AccessLog configuration takes an array.
# Each element of the array should be a two-element array where
# the first element is the stream (or anything responding to <<)  and
# the second element is the access log format.
# Please see webrick/accesslog.rb for available formats.

access_log_stream = File.open('/var/log/webrick/access.log', 'w')
access_log = [ [ access_log_stream, AccessLog::COMBINED_LOG_FORMAT ] ]

start_webrick(:Logger => server_logger, :AccessLog => access_log)
```

6 Hooks

WEBrick has many hooks you can tap into. Following is a flow-chart (somewhat) of the order of hook invocation.

```
Server:

  :ServerType.start (before yield)
  :StartCallback
    :AcceptCallback
    :RequestHandler
    # servlet invoked at this point
```

```
  :StopCallback
  :ServerType.start (after yield)


FileHandler Servlet:

  :DirectoryCallback or :FileCallback
  :HandlerCallback
  # handler is invoked at this point
```

7 HTTP Authentication

RFC 2617 specifies two mechanism for HTTP authentication: basic and digest. WEBrick supports both authentication mechanisms. WEBrick verifies authentication information against user-specified Apache-compatible user database.

Sometimes, you find that setting up a user database file troublesome. With basic authentication, you can pass a block of code to WEBrick that returns true if the authentication token is valid or false otherwise. This is a shortcut to having to create a user database file.

```
realm = "Gnome's realm"
start_webrick {|server|
  server.mount_proc('/convenient_basic_auth') {|req, resp|
    HTTPAuth.basic_auth(req, resp, realm) {|user, pass|
      # this block returns true if authentication token is valid
      user == 'gnome' && pass == 'supersecretpassword'
    }
    resp.body = "You are authenticated to see the super secret data\n"
  }
}
```

```
dede:~$ w3m -dump http://localhost:8080/convenient_basic_auth
Username for Gnome's realm: gnome
Password: supersecretpassword
You are authenticated to see the super secret data
```

7.1 Basic Authentication

Basic authentication is done by `HTTPAuth::BasicAuth`. If using a user database file, the file must be similar to what `htpasswd` (from Apache HTTP Server package) generates. The supplied `HTTPAuth::Htpasswd` parser can only understand passwords generated using the standard `crypt()` function. This means, you have to invoke `htpasswd` with the `-d` argument. On all platforms except Windows and TPF, `-d` is the default argument.

```
realm = "Gnome's realm"
```

```
start_webrick {|server|
  htpasswd = HTTPAuth::Htpasswd.new('/tmp/gnome.htpasswd')
  authenticator = HTTPAuth::BasicAuth.new(:UserDB => htpasswd, :Realm => realm)
  server.mount_proc('/htpasswd_auth') {|req, resp|
    authenticator.authenticate(req, resp)
    resp.body = "You are authenticated to see the super secret data\n"
  }
}

# -c create password file
# -d use the default crypt() function
# -b accept password specified on the command line

dede:~$ htpasswd -cdb /tmp/gnome.htpasswd gnome supersecretpassword
Adding password for user gnome

dede:~$ cat /tmp/gnome.htpasswd
gnome:O2.19saB33Yk.

dede:~$ w3m -dump http://localhost:8080/htpasswd_auth
Username for Gnome's realm: gnome
Password: notsosecretpassword
Wrong username or password
Username for Gnome's realm: gnome
Password: supersecretpassword
You are authenticated to see the super secret data
```

7.2 Digest Authentication

WEBrick requires a user database file for digest authentication. The file must be in a format similar to what `htdigest` produces. The parser for the file is `HTTPAuth::Htdigest`, and the authenticator is `HTTPAuth::DigestAuth`.

```
realm = "Gnome's realm"
start_webrick {|server|
  htdigest = HTTPAuth::Htdigest.new('/tmp/gnome.htdigest')
  authenticator = HTTPAuth::DigestAuth.new(:UserDB => htdigest, :Realm => realm)
  server.mount_proc('/htdigest_auth') {|req, resp|
    authenticator.authenticate(req, resp)
    resp.body = "You are authenticated to see the super secret data\n"
  }
}

dede:~$ htdigest -c /tmp/gnome.htdigest "Gnome's realm" gnome
Adding password for gnome in realm Gnome's realm.
New password: supersecretpassword
Re-type new password: supersecretpassword
```

```
dede:~$ cat /tmp/gnome.htdigest
gnome:Gnome's realm:97b64451958049b15eab578ecf5ea4b2

dede:~$ w3m -dump http://localhost:8080/htdigest_auth
Username for Gnome's realm: gnome
Password: supersecretpassword
You are authenticated to see the super secret data
```

8 Becoming a Proxy Server !NOT YET

9 Doing Virtual Host !NOT YET

Not ready yet. In the meantime, please see: the following post about virtual hosting with WEBrick.

10 Configuration Reference

10.1 Server Configuration

:ServerName Default: `Utils::getservername`, which usually outputs whatever value in `/etc/hostname`.

:BindAddress Default: `nil`. `"0.0.0.0"` and `"::"` have the same effect as `nil`, which is to listen to all available network interfaces. If you want WEBrick to listen to a particular network interface, give this the value of that network interface.

:Port Default: `80` (for `HTTPServer`). The listening port number. It can also take a string (typically a service name), which will be the resolved through `/etc/services` (or other OS-dependent mechanism) to port number.

:MaxClients Default: `100`. Maximum number of concurrent connections. WEBrick uses a new thread for each new connection. Thus, data in thread-local storage will be lost when the connection is closed.

:ServerType Default: `SimpleServer`. `SimpleServer` simply starts the server. This is provided mainly so that you can override how WEBrick starts the server, e.g.: provide starting and stopping hooks. Please see the Hooks section.

:Logger Default: `Log.new`. A simple logging library, implemented in `webrick/log.rb`. You may use another Log library, such as log4r.

:ServerSoftware Default: `"WEBrick/#{WEBrick::VERSION} (Ruby/#{RUBY_VERSION}/#{RUBY_RELEASE_DA`
For posterity purpose.

:TempDir Default: `ENV['TMPDIR']`—ENV['TMP']——ENV['TEMP']——'/tmp'—. Among the standard handlers, only `HTTPServlet::CGIHandler` uses this to capture the invoked cgi's stdout and stderr streams.

:DoNotListen Default: `false` which will cause WEBrick to listen on the `:BindAddress` at port `:Port`.

:StartCallback Default: `nil`. An alternative way to hook into the startup process. If not `nil`, the value must respond to `call` message. Please see the Hooks section.

:StopCallback Default: `nil`. Similar to `:StartCallback`, except called during the shutdown process.

:AcceptCallback Default: `nil`. Similar to other callbacks, but called when a new connection has been accepted. The socket of the accepted connection is passed as the argument.

:RequestTimeout Default: `30` (seconds). Specifies how long to wait for each read operation on the socket. Some reads are line-based, for example, while reading the request-line, the headers, and chunked body; while some are stream-based

:HTTPVersion Default: `HTTPVersion.new("1.1")`. If WEBrick receives a non-HTTP 1.1 request, it will responding appropriately by using whatever HTTP protocol the request specify.

:AccessLog Default: `[[ $stderr, AccessLog::COMMON_LOG_FORMAT ], [ $stderr, AccessLog::REFERER` Please see the Logging section for further description.

:MimeTypes Default: `HTTPUtils::DefaultMimeTypes`. Please see Overriding Default MIME Type section

:DirectoryIndex Default: `["index.html","index.htm","index.cgi","index.rhtml"]`. `FileHandlers` look for these files when it receives a request for displaying a directory. If it finds any of these files, the file will be displayed instead of a file listing of the directory.

:DocumentRoot Default: `nil`. If it is not `nil`, WEBrick will setup a `FileHandler` for request-URI '/' to the specified filesystem path. Please see FileHandler section.

:DocumentRootOptions Default: :FancyIndexing =¿ true . Please see FileHandler Config Reference for other options.

:RequestHandler Default: `nil`. If not nil, it will be invoked like so: `handler.call(request, response)` before WEBrick services the request. Please see the Hooks section.

:ProxyAuthProc !NOT YET Default: `nil`.

:ProxyContentHandler !NOT YET Default: `nil`.

:ProxyVia !NOT YET Default: `true`.

:ProxyTimeout !NOT YET Default: `true`.

:ProxyURI !NOT YET Default: `nil`.

:CGIInterpreter !NOT YET Default: `nil`.

:CGIPathEnv !NOT YET Default: `nil`. !NOT YET.

:Escape8bitURI !NOT YET Default: `false`. !NOT YET. need more detailed explanation. If `true`, then escape 8-bit characters in request-URI contains 8-bit before parsing it.

10.2 FileHandler Configuration

:NondisclosureName Default: `".ht*"`. In a directory listing, any any file that matches the value (as per shell-globbing, not regular expression) is not displayed. If the request-URI refers to a file that matches the value, `FileHandler` will return a 403 (Forbidden) status.

:FancyIndexing Default: `false`. If this is `true` and the request-URI refers to a directory, and not a file, then `FileHandler` servlet will list the contents of that directory. Otherwise, it will return a 403 (Forbidden) status.

:HandlerTable Default: `{}`. This is a mapping of filename suffix =¿ handler. If this is left blank, then all request for file is passed on to an instance of `HTTPServlet::DefaultFileHandler`. This handler understands the HTTP's range directive (partial file transfer).

:HandlerCallback Default: `nil`. A callback which is invoked before the handler for the request.

:DirectoryCallback Default: `nil`. A callback which is invoked before the handler for the request (and before `HandlerCallback`) if the request-URI refers to a directory.

:FileCallback Default: `nil`. Similar to `DirectoryCallback` except if the request-URI refers to a directory.

:UserDir Default: `"public_html"`. If the `FileHandler` servlet is mounted on `'/'`, and the request-URI starts with `'/~username'`, then it is mapped to `"#{username's home dir}/#{:UserDir value}"`.

10.3 BasicAuth Configuration

:UserDB An instance of `HTTPAuth::Htpasswd` initialised with the filename of the htpasswd file.

:Realm You have to supply this, but it is not used.

10.4 DigestAuth Configuration

:UserDB An instance of `HTTPAuth::Htpasswd` initialised with the filename of the htpasswd file.

:Realm You have to supply this, and it is used.

11 Class&Module Reference

11.1 HTTPRequest

Following is a list of methods of a `HTTPRequest` object. The list also contains example values corresponding to this HTTP request:

```
GET /foo/bar?key1=value1&KEY2=value2 HTTP/1.1
Host: localhost:8080
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9
Accept: text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Request line

request_line `"GET /foo/bar?key1=value1&KEY2=value2 HTTP/1.1\r\n"`

request_method `"GET"`

unparsed_uri `/foo/bar?key1=value1&KEY2=value2`

http_version `HTTPVersion.new("1.1")`. If the request line is missing the HTTP part, it is considered to be HTTP 0.9.

Request-URI

request_uri `::URI::parse("http://localhost:8080/foo/bar?key1=value1&KEY2=value2")`

host `"localhost"`

port `"port"`

path `"/foo/bar"`

query_string `key1=value1&KEY2=value2`

script_name `"/foo"`

path_info `"/bar"`

Header and Entity Body

raw_header `[ "Host: localhost:8080\r\n",`
`"Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9\r\n",`
`"Accept: text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1\r\n",`
`"Accept-Encoding: gzip,deflate\r\n",`
`"Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n",`
`"Keep-Alive: 300\r\n",`
`"Connection: keep-alive\r\n" ]`

header a hash of key =¿ [value] of the header. The header name is downcased. If there are multiple header names, their values are appended to the array of values.

[] The result of `request.['ACCEPT']` is `request.header['ACCEPT'.downcase].join(", ")`

each Invoke the passed block for every header key-value pair, e.g.: `req.each {|key, value| puts "#{key} =`

keep_alive `true`. Set true if the header `'connection'` is not set to 'close' and the request is in HTTP 1.1.

keep_alive? `true`. Alias of `keep_alive`.

cookies An array containing instances of `Cookie`, each representing a cookie that the client sent. These cookies are not automatically copied to the `HTTPResponse` object.

query `{"key1"=>"value1", "KEY2"=>"value2"}`. This is a table of `key => value`. `value` is of type `FormData` which is just a subclass of `String`. This information matters only if you have duplicate keys in the `query_string`.

body a String containing the body of the request. It is `nil` unless the request is POST or PUT.

Miscellaneous

user `nil`. This is set if the client is using HTTP authentication.

addr `["AF_INET", 8080, "dede", "127.0.0.1"]`. The local address of the socket on which this request is received.

peeraddr `["AF_INET", 37934, "dede", "127.0.0.1"]`. The address of the client.

attributes `{}`. I am not sure what this is for.

request_time a Time object, set to when the request is made.

meta_vars a hash filled containing the CGI meta-variables. The CGI specification has a list of these meta-variables.

11.2 HTTPUtils::FormData

The `FormData` object is a subclass of `String`. It is used to represent query values. In a query, the same key may be assigned multiple values. Each value is assigned to an instance of `FormData`. This instance stores a reference to the next instance of `FormData` that stores the next value, and so on.

each_data Pass a block to it and for each value, it will call the block.

list Puts the values into an array.

11.3 HTTPResponse Object

Many of the methods in `HTTPResponse` are called by WEBrick after your servlet has serviced the request. Instead of listing all public methods as in the `HTTPRequest` listing above, the following only lists methods that is meaningful in servlet context:

status= You can set the response status using this, e.g.: resp.status = 202

[]= You can set a custom header using this, e.g.: resp['content-type'] = 'text/html'

body= You can set the body of the response using this. It can also be an IO object in which case, the content is transmitted in blocks.

set_redirect Sends a redirect response to the given URI, e.g.: `resp.set_redirect(HTTPStatus::MovedPerma`

cookies An array containing instances of `Cookie` that are going to be sent back to the client. Initially the array is empty as the cookies received from the clients are not automatically copied here.

11.4 Cookie

name The name of the cookie. The name of the cookie may only be read, not set

value The value of the cookie. value should be in a printable ASCII encoding.

version Identifies which cookie specification this cookie conforms to. 0, the default for Netscape Cookies, and 1 for RFC 2109 cookies.

domain The domain for which the cookie is valid. An explicitly specified domain must always start with a dot.

expires A Time or String representing when the cookie should expire. Expires must to be in the following format: `Wdy, DD-Mon-YYYY HH:MM:SS  GMT`

max_age The lifetime of the cookie in seconds from the time the cookie is sent. A zero value means the cookie should be discarded immediately.

comment Allows an origin server to document its intended use of a cookie. The user can inspect the information to decide whether to initiate or continue a session with this cookie.

path The subset of URLs to which the cookie applies.

secure When set to true, the cookie should only be sent back over a secure connection.

11.5 HTTPStatus Module

Parent Class	Response Code	Class Name
Info	100	Continue
	101	SwitchingProtocols
Success	200	OK
	201	Created
	202	Accepted
	203	NonAuthoritativeInformation
	204	NoContent
	205	ResetContent
	206	PartialContent
Redirect	300	MultipleChoices
	301	MovedPermanently
	302	Found
	303	SeeOther
	304	NotModified
	305	UseProxy
	307	TemporaryRedirect
ClientError	400	BadRequest
	401	Unauthorized
	402	PaymentRequired
	403	Forbidden
	404	NotFound
	405	MethodNotAllowed
	406	NotAcceptable
	407	ProxyAuthenticationRequired
	408	RequestTimeout
	409	Conflict
	410	Gone
	411	LengthRequired
	412	PreconditionFailed
	413	RequestEntityTooLarge
	414	RequestURITooLarge
	415	UnsupportedMediaType
	416	RequestRangeNotSatisfiable
	417	ExpectationFailed
ServerError	500	InternalServerError
	501	NotImplemented
	502	BadGateway
	503	ServiceUnavailable
	504	GatewayTimeout
	505	HTTPVersionNotSupported

12 Glossary

Callback An object that respond to the `call` message. Usually this is an instance of Proc or Method.

Path-Info The trailing path after the handler's path. If a handler is mounted at '/foo', and the request-URI is '/foo/bar/is/boring', then path-info would be '/bar/is/boring'

Request-URI the path specified in a HTTP URI. For example, the request-URI of 'http://hoohoo.ncsa.uiuc.edu/cgi/env.html' is '/cgi/env.html'

13 Author's Note

Author's Note

The first WEBrick-based application I built was a port of a Java REST-ful server. I attended a seattle.rb meeting where Eric Hodel was demonstrating WEBrick. At that time, I was a bit overwhelmed maintaining a Java-Servlet-based REST-ful server due to the extensive class hierarchy (there were 560-ish classes). Many of them are used to get around Java restrictiveness, for example, for creating first-class function object (Proc or block in Ruby).

The performance I am getting is also acceptable, averaging 50 requests/second on a 600MHz P-III machine 256MB, a bit faster than Tomcat's 40 requests/second (I suspect because of lighter memory requirement which translate to less frequent swapping on that machine). The memory usage is also acceptable, hovering around 27 MB for about 100 concurrent client compared to 127MB in Tomcat. Yes, I probably should not have been using Tomcat as comparison as it is well-known to be a behemoth, but that is the official Java Servlet container and also the most widely-used too.

Obviously, this statistics are very activity-dependent. A simple hello world server would be chastised for having this statistics.

In any case, I hope you will enjoy using WEBrick; I certainly do.

14 License

15 GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary

Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which

states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network

location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the

original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU

> Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.