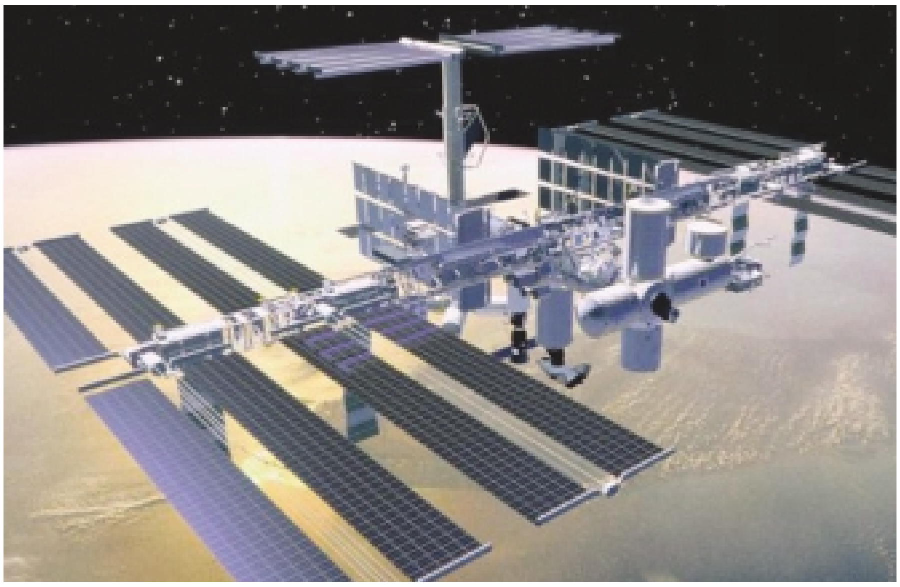
A Small Selection of Useful Apache Modules

Docking Station

There are literally hundreds of modules for the Apache web server. And that makes it difficult to pick out the goodies, or even find the right software for the job in hand. This is the starting point for a compact overview of the authors' favorite modules. BY STEFAN WINTERMEYER, RONALD KARSTENS

A web server is a simple piece of software when you think about it. Reading any HTML file from the file system and serving it up to the browser is a task that a comptetent programmer could probably handle with a few hundred lines of code. But of course a good web server like Apache can do a whole lot more. And modules are the source of most of the web server's intelligence.

The Apache web server has gone through a similar evolution to the Linux Kernel in this respect. It used to be monolithic, but now more and more functionality is provided by modules. This is particularly evident in the new Apache 2.0 version, although there are still probably more modules available for Apache 1.3 than the authors have had hot dinners [1]. Having said that, a handful of practical modules can make a web admin's daily grind a whole lot easier.



MPM: Multiprocessing Modules in Apache 2.0

Apache 2.0 sees the introduction of a new group of modules: the server delegates basic network and operating system functionality to so-called multiprocessing modules (MPMs). The MPMs in turn allow Apache to process multiple requests simultaneously. The admin has several options to choose from – but changing options is simple if they are modular. MPMs are discussed in detail in the article on page 29.

Mod_speling: "Spel"checking for URLs

Good web servers provide a structure that allows users to access some content directly via URLs. Sometimes users will more or less know the URL, although they are not sure about the case. Mod_speling can help you solve this issue. As you might have guessed from the way the module misspells "spelling", this module corrects typos.

If there is a small typo in the URL a web user provides, instead of issuing an error message the user will be pointed to the right page. The module will correct up to one mistyped character and any number of lower/upper case errors. This will save most web users headaches or yet another visit to Google. Take a look at [2] for more information on mod_speling.

Mod_include: SSI – Server Side Includes

When the first frame based web pages were introduced, some webmasters ironically suggested that their colleagues possibly needed so many frames because they had never heard of server side includes (SSIs). And there is certainly a modicum of truth in this statement: If you want to use the same header, a footer and a menu from the current directory without frames, there is nothing to prevent you using native tools and ditching the frames, and the SSI "include" element is the way to do this.

SSI instructions are directly embedded in special tags in HTML documents. "<!--#include virtual="*File*" -->" will cause SSI to read the content of a file at the tag position before the server serves up the page.

Most standard distributions use an Apache configuration that passes "*.shtml" files to the SSI module ("mod_include"). If this does not work straight away, a quick look at your "httpd.conf" may provide some clues. You will need the following entry:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

The "Options +Includes" in the "<Directory>" section is used to enable server side includes. SSI can also be used recursively, as you can see from Listing 1 and 2. In line 11 Listing 1 integrates the "/includes/footer.shtml" file. This file is also parsed by the SSI module and will output the current date in about line 4.

If you need to, you can use SSIs to run commands on the web server and

Listing 1: Example of SSI

```
<html>
  <body>
    <!--#include virtual="/includes/header.html" -->

    <table><tr><td>
      <!--#include virtual="menue.html" -->
    </td><td>
      hello world!
    </td></tr></table>

    <!--#include virtual="/includes/footer.shtml" -->
  </body>
</html>
```

include the result. The following instruction adds the output from "ls" to a web page:

```
<!--#exec cmd="ls" -->
```

External commands can take so long to execute that the page build up is either extremely slow, or does not happen at all (timout). But otherwise there is no real reason to expect SSIs to cause performance bottlenecks on large web sites. It does not always make sense to enable SSI for the ".html" and ".htm" suffixes, but if you do, your web server should handle the situation gracefully.

If users other than the webmaster can create HTML files, "*#exec*" can open up a large security hole, allowing normal users to execute commands with the web server's privileges. You will need to apply the "Option +IncludesNOEXEC" configuration, to close the gap in this scenario. SSIs will still be enabled, but dangerous commands will not. For more information and examples on SSIs see [3] and [4].

Mod_php: Server Side Scripting

If SSIs do not provide sufficient functionality for you, you might like to try PHP as a universal tool for dynamic web pages. The server will first parse the page at runtime and then serve up the normal HTML files. Mod_php will handle any files with the "*.php" suffix.

Listing 3 contains a Hello World program in PHP. As you can see in line 3, this module also uses special tags to include commands in HTML files: "<?php *command* ?>". As PHP is a fully-fledged scripting language with various database interfaces, there are no limits to what you can do.

Installing PHP is quite complex, as you need to enable a variety of add-ins and options prior to this step. This applies equally to dynamic libraries and PHP modules, and to the question as to the database interfaces you will need. We recommend compiling PHP as a DSO module (Dynamic Shared Object), as you can then replace PHP without re-installing your Apache.

Although the documentation still has not caught up to Apache 2.0, the current PHP versions seem to work well with the new server. The following example of a "configure" call uses MySQL as its database backend:

```
$ CC=gcc CFLAGS=-O2 LDFLAGS=-s ↵
./configure ↵
--with-mysql=/opt/mysql ↵
--with-apxs2=/opt/apache/bin/↵
apxs --prefix=/opt/apache ↵
--sysconfdir=/etc/php
```

Mod_perl: Embedding Scripting Languages in HTML

We should not forget Perl while we are discussing PHP. Dynamic websites in Perl do not require any Apache modules, as you can write CGI programs in Perl. This variant does have its disadvantages, as it means launching a Perl instance every time the page is opened. A new process is created, and the whole Perl interpreter is loaded into virtual memory where it can translate and run the CGI program. Of course this works, but it is slow and does not scale well.

One possible solution is to use an Apache module. When you launch your Apache, mod_perl loads a single instance of Perl which stays active and handles any requests for Perl programs. This assumes that you included mod_perl when compiling the web server, or loaded the DSO module – and this is no trivial task. Some CGI programs also require you to perform modifications, and you will not be able to harness the full power of mod_perl without adapting your scripts.

See [6] for a howto and some useful tips. Warning: Apache 2.0 requires a mod_perl version from the 2.0 series, which is currently beta – this is one reason why some production servers are still hesitating before moving up to Apache 2.0.

Mod_bandwidth: Less Bandwidth is More

Most webmasters are familiar with this dilemma: You have been asked to place a video or an MP3 file by an up and coming artist on your web server – but without using up all your bandwidth. You can use a bandwidth limiter to do so. Most Linux distributions do not provide a module for this task and to make things worse there are several different approaches. You can refer to [7], using the "bandwidth" search key for a list of available modules.

If you are experiencing general bandwidth problems, you might like to

Listing 2: "footer.shtml"

```
<!-- begin footer -->
  <HR>
  <!--#config timefmt="%A %B %d, %Y" -->
  Today is <!--#echo var="DATE_LOCAL" -->
  <BR>
  <!--#config timefmt="%D" -->
  This file last modified <!--#echo var="LAST_MODIFIED" -->
<!-- end footer -->
```

Listing 3: "hello-world.php"

```
<html>
<body>
<?php echo "Hello World<p>"; ?>
</body>
</html>
```

refer to the section on Traffic Shaper [9] in the networking howto. But be careful if you impose bandwidth restrictions, just a slight error can severely impact your web server's performance.

Mod_mp3: Simple MP3 Streaming

With bandwidth restrictions and the MP3 keyword, the obvious place to go would be an MP3 server of your own. And again Apache can offer you a module for this task, allowing you to provide a constant stream of music to your MP3 clients, such as XMMS [10], instead of simply allowing shared access to audio files.

If you want to run your own radio station on your intranet (with copyright-free music, of course), you should definitely take a look at mod_mp3 streaming [11]. Again, most distributions do not include this module and you will need to compile and include in your Apache implementation.

The configuration steps are simple. In Listing 4 the module is used to serve up the MP3 files in "/home/export/mp3", and his two favorite songs (lines 7 through 9) at random (line 10) as "Stefan's Radio" (line 4).

Listing 4: MP3-Streaming-Server

```
<VirtualHost mp3.example.com:⮯
8000>
  ServerName mp3.example.com
  MP3Engine On
  MP3CastName "Stefan's Radio"
  MP3Genre "European Trance"
  MP3 /home/export/mp3
  MP3 /tmp/favesong1.mp3
  MP3 /tmp/favesong2.mp3
  MP3Random On
  #Increase this if your ⮯
connections are timing out
  Timeout 1200
  ErrorLog /var/log/httpd/⮯
music-stream_error_log
</VirtualHost>
```

Mod_auth_ldap: Using LDAP for User Authentification

Most web sites have restricted areas intended only for specific users. External web servers can use the "mod_auth" mechanism [12], to handle this requirement. The user credentials are stored in a password file on the web server.

But the demand for single sign-on for Intranet web servers continues to increase, failing that you may be asked to at least ensure that the same password can be used for the whole range of services on offer. Centralized password management on a Novell, Microsoft, or Open LDAP server [13] is quite common, but there is no reason for Apache to hide its light under a bushel. Mod_auth_ldap [14] uses LDAP for user authentification. Listing 5 shows a configuration example.

Mod_ssl: Serving Up Web Pages with SSL and TLS

Encrypted communication is becoming increasingly important and this development is reflected by Apache. Only a few months ago integrating encrypting modules used to be a fairly complicated and time-consuming task. However, Apache 2.0 comes pre-configured with mod_ssl [15]. Ensure that you have the OpenSSL libraries (which should be installed by default on any current Linux distribution), and then type "./configure --enable-ssl" to compile Apache with SSL in place.

A production HTTPS server requires a cryptographic certificate. Depending on your target group, you may decide to approach a commercial Certificate Authority or set up your own CA. The certificate and key files will then be stored below the Apache configuration directory and read on launching the server after, possibly, prompting you for a password. ■

Listing 5: LDAP Authentification

```
<Directory "/usr/local/http/⮯
htdocs/a-team-doku">
 Options Indexes FollowSymLinks
 AllowOverride None
 order allow,deny
 allow from all
 AuthName "A-Team only"
 AuthType Basic
 LDAP_Server ldap.example.com
 LDAP_Port 389
 Base_DN "o=A-Team HQ,c=DE"
 #Bind_Pass "secret"
 UID_Attr uid
 require valid-user
</Directory>
```

Info

[1] Overview of Apache Modules: *http://modules.apache.org/*

[2] Typo Correction: *http://httpd.apache.org/docs/mod/mod_speling.html*

[3] Server Side Includes: *http://httpd.apache.org/docs/mod/mod_include.html*

[4] SSI Howto: *http://httpd.apache.org/docs/howto/ssi.html*

[5] PHP Tutorial: *http://www.php.net/manual/en/tutorial.php*

[6] Mod_perl Download and Documentation: *http://perl.apache.org/*

[7] Search for Apache Modules: *http://modules.apache.org/search*

[8] Docs for mod_bandwidth: *http://www.cohprog.com/v3/bandwidth/doc-en.html*

[9] Info for Traffic Shaper: *http://www.tldp.org/HOWTO/Net-HOWTO/x1416.html*

[10] XMMS: *http://www.xmms.org*

[11] Mod_mp3: *http://media.tangent.org/*

[12] User Authentification: *http://httpd.apache.org/docs/howto/auth.html*

[13] OpenLDAP: *http://www.openldap.org/*

[14] Mod_auth_ldap Homepage: *http://www.muquit.com/muquit/software/mod_auth_ldap/mod_auth_ldap.html*

[15] Introduction to SSL and TLS on Apache: *http://httpd.apache.org/docs-2.0/ssl/*

[16] Apache 2.0 Module Documentation: *http://httpd.apache.org/docs-2.0/mod/*

[17] Apache and Various Modules: *http://www.apachetoolbox.com/*