Caudium HOWTO

David Gourdelier

Caudium Group

vida at caudium dot net

September 2002

Revision History

Revision 2.2	2002−09−20	Revised by: dg
Few more fixes, document now contains images and icons. See revision history for more information.		
Revision 2.1	2002−06−14	Revised by: dg
Document is now well−formed XML, one more example. See revision history for more information.		
Revision 2.0	2002−06−14	Revised by: Tab
Converted to DocBook XML 4.1.2. See revision history for more information.		
Revision 1.0	2002−06−06	Revised by: dg
Initial release		
Revision 0.91		
Indentation of the SGML document and one more paragraph. See revision history for more.		
Revision 0.9		
This Howto is going into stable state. See the revision history for more information.		
Revision 0.05	2002−04−04	
This HOWTO is in the DocBook format. See the revision history for more information.		
Revision 0.04		
Added Introduction and license. See the revision history for more information.		
Revision 0.03		
Added some useful corrections. See the revision history for more information.		
Revision 0.02		
Minor correction in other answers. See the revision history for more information.		
Revision 0.01	2002−03−11	
Initial revision. See the revision history for more information.		

This is the Caudium HOWTO. This document will give an overview of the Caudium server. New users will learn how to set a basic virtual server up. More experienced users will find useful tips on development and

optimization. Finally at the end I give some documentation on Caudium top priorities for people who want to contribute.

Table of Contents

Table of Contents

Chapter 1. License

This document is a volunteer effort. Feel free to improve and make changes. If you catch any mistakes, please correct them.

Chapter 2. Overview

2.1. What is Caudium?

Caudium is a Web server based on a fork of the Roxen Challenger 1.3 WebServer. Like Roxen, Caudium is written in Pike with parts written in C for performance reasons. Pike is an interpreted language developed by Frederik Hübinette and Roxen Internet Software (RIS), a Swedish company which also created the Roxen Web Server. Caudium, like Pike, is distributed under the terms of the GPL license; several companies and people are involved in its development.

Caudium features include:

- Single−process architecture.
- Optional multi−threaded mode.
- Backwards compatible with Roxen 1.3 on the API and RXML level.
- Runs on many Unix−like systems (GNU/Linux, FreeBSD, OpenBSD, NetBSD, Solaris, AIX, Darwin/MacOS X).
- Web based interface for easy administration.
- Built in SSL capabilities. Enabling SSL on Caudium is as easy as filing a web form.
- Written in Pike. Unlike most web servers, you don't need to learn C or C++ and those pesky details like memory allocation to enhance the server's capabilities.
- Extensible with custom modules.
- Powerful API.
- Lots of standard modules distributed with the server, including an FTP server, the CAMAS web mail application, and the UltraLog log analysis tool. The "module" directory in Caudium 1.2 with CAMAS contains 192 code−only files. Most of the modules are in a single file.
- RXML language. RXML stands for Roxen eXtensible Markup Language and is a set of tags, containers and simple programming language constructs that you put in your HTML source. Those tags and containers will be interpreted at run−time by Caudium. This allows non−programmers to do development.
- XML language. Using Sablotron, Caudium can render XML pages processed with XSLT, DOM and XPath. You can find more information on Sablotron on http://www.gingerall.com/. It allows you to use pages designed for Apache mod_xslt in Caudium without modifications. Also the output of these generated pages can also be parsed by our RXML parser before sending the result to the client.

2.2. Caudium vs Roxen: Why a fork?

Caudium is backward compatible with Roxen Challenger 1.3. It has all the Roxen 1.3 RXML tags and the Roxen 1.3 API, so, from a technical point of view, it is quite the same (although Caudium has many improvements over Roxen 1.3). The main legal difference between Roxen 1.3 and Caudium is that Roxen is property of Roxen Internet Software whereas Caudium is owned by its developers, The Caudium Group. There are many functional extensions, improvements and bug fixes over the original Roxen 1.3 code base from which Caudium was created. Here you may say it is not a problem since it is under GPL. The problem is that even it is licensed under the GPL, RIS may not include your patch in its CVS tree, you may not have a CVS account, so you can't really do all you want. The main reason for the fork, however, was that RIS didn't pay attention to the users' needs, and proceeded to make the new versions of the Roxen Web Server largely incompatible with the previous versions, to the point where one couldn't switch to the new version of the software without much effort put into conversion of the old source code. Rewriting huge portions of RXML/Pike code just to switch to a new version of the web server seemed to be quite counter−productive, and thus the Caudium Project was born.

2.3. Comparing Caudium with Apache

Caudium differs from Apache in many ways including the directory structure, programming language, and type of configuration. Caudium has a fully integrated web interface, while Apache relies on editing text files directly. Moreover, any change to the Apache configuration requires a restart of the server, while Caudium sees the changes immediately after you save them from the web interface. There are also some differences in the vocabulary used in the configuration interface. Another difference is that Apache 1.3 uses a forked process model while Caudium uses threads or a monolithic process model, depending on the features present in the copy of Pike it uses. Caudium allows the programmer/user to easily extend the server using modules/scripts written in Pike that integrate tightly with the core server and, thus, create a more robust, faster and more intuitive entity than an Apache server augmented with a set of external dynamically loadable libraries. While changing anything in the source code of any Apache extension (or the server itself) requires recompilation, relinking and restarting of the whole server, Caudium allows one to add/remove/modify code without any interruption of the server operation. If you modify a module all you need to do to see the effects of your work is to reload the module in question using the configuration interface. Caudium's architecture also provides easy means of building highly dynamic web pages that use SQL, gdbm, Mird databases, LDAP, dynamically generated graphics (including business graphics modules) and more. The same effect can be achieved with Apache almost only by using external set of Perl, Python, or other language, scripts running as CGI or embedded using a dynamically loaded module. While allowing the programmer/designer to use CGI modules, Caudium offers more power when the source code is integrated with the Caudium core.

2.4. Advantages of Caudium

- Stable multi−threaded programming (Roxen has been multi−threaded since 1994).
- Mostly written in an interpreted, object−oriented, language − Pike:
    - Easy to program and extend. Just imagine if you could code extensions to the Apache core in PHP or Perl.
    - No buffer overflows, segmentation faults or any other problems of that nature.
    - RealNetworks is still using Roxen 1.3 as their main web platform for its stability and speed.
- Low level API available for end−users.
- RXML tags.
- Lots of modules.
- Backwards compatible with Roxen 1.3.
- Different. If you are a Linux or BSD user for some time you already know the most used is not always the best.
- Caudium won't run under win32 (thus letting us more time for useful things). However some people are trying to port it using cygwin without code modification.
- Embedded PHP4 support.

2.5. Disadvantages of Caudium

- Too small of a community, and quite unknown in web server world. One example of the problem you may have is if you write Pike code for Caudium, and your hosting company doesn't support Pike, and you don't have root access to your server. Of course, it is possible to compile Pike on your account and run Caudium without the superuser privileges.
- RIS (Roxen Internet Software) owns Pike, but this is likely to change in the not−so−distant future.

- Needs more testing for exotic modules or modules we don't use often. Example of such modules include htaccess support.
- Too little documentation. One example is the lack of documentation about FastCGI support.
- CGI performance on busy CGI based sites was lackluster –– no one will take the time to rewrite CGIs. This seems to be somewhat resolved with Pike 7.2/Caudium 1.2.
- It is sometimes impossible to get some features when you are a normal user because configuration is only accessible with the CIF. (Configuration InterFace). And the CIF. is accessible only by the admin (this is being taken care of in the development series of the server). Here is list of day to day problems submitted by Chris Davies:

  > Webmasters like to use mod_rewrite in the .htaccess files to protect their sites from leeching, etc. Yes, there is referrerdeny and browserdeny, but these are accessible to the System Administrator, not the casual web user. On Apache, one can easily specify rules on a per directory basis. This isn't easy in Caudium, however, it is rare that someone would have more than two or three different conditions.
  >
  > Now, for people that are running their own server, many of these issues disappear. Administration and adding servers is much easier. I'm not sure I agree with the templates, but perhaps an Apache Template could be tweaking beyond that to the System Administrator. In most cases, Apache has SSI turned on for shtml files –– getting the exec command enabled to mimic this in Caudium is not a task for the beginner. The maze of options to find that one setting will certainly frustrate anyone getting started.
- A graphical interface is not always the best. To create a virtual server with Apache, you have to copy and paste five lines. If you use the Caudium Interface it will take you more than one minute. And if you want to change Caudium text file directly, you will have to write sed script.

Chapter 3. Getting packages

3.1. How to get packages for Debian GNU/Linux

In Debian GNU/Linux Woody, also known as Debian 3.0, Caudium and Pike 7.0 and 7.2 are included in the distribution. If you use Woody, you will have Caudium 1.0 and Pike 7.0/7.2. If you use Sid you will have the latest Caudium and the latest Pike; these two packages are actively maintained by the Caudium community, and you will have frequent updates. If you want to test the latest Caudium under Woody, add this to your /etc/apt/sources.list:

```
deb http://grendel.firewall.com/debian/caudium/1.2 unstable main
```

This location is mirrored in other locations, so you can also use:

```
deb http://ftp.oav.net/caudium unstable main
deb ftp://ftp.oav.net/caudium unstable main
```

If you want php4 Debian package for Caudium, you can do:

```
# su -
# apt-get update
# apt-get install -f caudium caudium-modules \
  caudium-ultralog caudium-pixsl caudium-dev
```

And then grab the PHP4 sources:

```
# exit
$ cd /usr/src; apt-get source -d php4; apt-get build-dep php4
```

If you don't have devscripts and debhelper installed, install them now. Then enter the /usr/src/php–4* directory and do:

```
$ debuild binary
```

After some time and some coffee you should have php4 packages for Apache, CGI and your fresh Caudium.

3.2. How to get packages for FreeBSD

Caudium 1.0 and Pike 7.0 are in the official FreeBSD ports. There are also available as a binary package in the FreeBSD 4.7 CDROMs.

Caudium 1.2 and Pike 7.2 are available as ports tarballs at ftp://ftp.oav.net/pkg_freebsd/ports.tar.gz. If you take the latest ports from this site use the following command:

```
$ cd /tmp
$ fetch ftp://ftp.oav.net/pkg_freebsd/ports.tar.gz
Receiving ports.tar.gz (32468 bytes): 100%
```

```
32468 bytes transferred in 0.6 seconds (53.87 kBps)
$ su -
# cd /usr/
# tar xzf /tmp/ports.tar.gz
```

Another note, since Caudium and Pike have been added recently to FreeBSD port tree, you must have an up to date port tree. See the FreeBSD handbook at http://www.freebsd.org/handbook/cvsup.html about cvsup handling.

As root, go to `/usr/ports/www/caudium10` to install current stable version, `/usr/ports/www/caudium12` to install the next stable version in Release Candidate process, or `/usr/ports/www/caudium-dev` to install the current developer's version. The script will automatically fetch and install Pike, try to detect what library you've installed and compile all the necessary libs for Caudium and Pike.

Here is the current Pike / Caudium options supported by the current port :

- `WITH_MOST` : Install the FreeType1 and FreeType2 support.
- `WITH_OPTIMIZED_CFLAGS` : Add some performance flags to the compiler.
- `WITH_TTF` : Install the FreeType1 support.
- `WITH_FREETYPE` : Install the FreeType2 support.
- `WITH_MYSQL` : Install the MySQL support.
- `WITH_POSTGRES` : Install the PostgreSQL support.
- `WITH_MSQL` : Install mSQL support.
- `WITH_UNIXODBC` : Install UnixODBC support (incompatible option with `WITH_IODBC`).
- `WITH_IODBC` : Install iODBC support (incompatible option with `WITH_UNIXODBC`).
- `WITH_SANE` : Install SANE backend.
- `WITH_PDF` : Install PDF support
- `WITH_MESA`/`HAVE_MESA` : Install MesaGL support.
- `HAVE_GNOME`/`WANT_GNOME` : Install GNOME support.

There are some options related only to Pike, like the GNOME / Mesa ones that are uninteresting to Caudium, but exist for Pike support.

A little example on how to install Caudium on FreeBSD :

```
# cd /usr/ports/www/caudium12
# make WITH_MOST=yes WITH_MYSQL=yes WITH_OPTIMIZED_CFLAGS=yes install clean
===>  Extracting for caudium-1.2.6
>> Checksum OK for caudium-1.2.6.tar.gz.
===>   caudium-1.2.6 depends on executable: pike - found
===>   caudium-1.2.6 depends on executable: gmake - found
===>   caudium-1.2.6 depends on shared library: sablot.67 - found
[...]
===>   Generating temporary packing list
***************************************************************
  If this is the first installation of Caudium, please go the
  caudium's directory and execute the server/install script
  to finish the Caudium installation.

  If your are upgrading, just start caudium as usual.

  NOTE: there is an automatic starting script in etc/rc.d/
```

```
************************************************************
===>   Registering installation for caudium-1.2.6
===>  SECURITY NOTE:
      This port has installed the following startup scripts which may cause
      network services to be started at boot time.
/usr/local/etc/rc.d/caudium.sh.sample

      If there are vulnerabilities in these programs there may be a security
      risk to the system. FreeBSD makes no guarantee about the security of
      ports included in the Ports Collection. Please type 'make deinstall'
      to deinstall the port if this is a concern.

      For more information, and contact details about the security
      status of this software, see the following webpage:
http://caudium.net/
===>  Cleaning for libiconv-1.7_5
===>  Cleaning for gdbm-1.8.0
===>  Cleaning for mird-1.0.7
===>  Cleaning for mysql-client-3.23.49
===>  Cleaning for autoconf-2.53
===>  Cleaning for autoconf213-2.13.000227_1
===>  Cleaning for automake14-1.4.5
===>  Cleaning for bison-1.35_1
===>  Cleaning for gettext-0.11.1_3
===>  Cleaning for gmake-3.79.1_1
===>  Cleaning for libtool-1.3.4_3
===>  Cleaning for m4-1.4_1
===>  Cleaning for nasm-0.98,1
===>  Cleaning for jpeg-6b_1
===>  Cleaning for tiff-3.5.7
===>  Cleaning for pexts-20020121
===>  Cleaning for pike72cvs-7.2.356_5
===>  Cleaning for libgmp-4.0.1
===>  Cleaning for freetype-1.3.1_2
===>  Cleaning for freetype2-2.0.9
===>  Cleaning for libmcrypt-2.5.0
===>  Cleaning for mhash-0.8.14
===>  Cleaning for expat-1.95.2
===>  Cleaning for Sablot-0.81_1
===>  Cleaning for caudium-1.2.6
#
```

Now if you need to install PHP4 support for Caudium, just go to `/usr/ports/www/caudium_php4` and do a make install clean as well.

3.3. How to get packages for Solaris

Bill Welliver has made some packages. They are available on Riverweb and mirrored to Oav.net.

3.4. How to get CAMAS from CVS/source

To get CAMAS from the CVS, go to the CAMAS web site and follow the link.

Chapter 4. Creating your first server

4.1. Installing Caudium from sources

You need to install Pike before you can install Caudium. The version of Pike is different depending on the Caudium version you want to install. For Caudium 1.0, use Pike 7.0; for Caudium 1.2, use Pike 7.2. You can find some Pike packages for the most popular systems: Debian GNU/Linux (Woody/Sid), FreeBSD, Solaris.

To take a CVS source snapshot of Pike 7.2+, you will need to have a preinstalled Pike (any version) on your system in order to compile the fresh CVS checkout. This is because the CVS sources need to pre–process certain parts of the sources with some special Pike scripts. Downloading a tarball from the RIS site (always outdated) will free you from the requirement of having the Pike installed before compiling the CVS snapshot.

For more information, see the quick start guide at http://caudium.net/.

4.2. Starting

The first time you install Caudium from the sources, you will need to type the following commands:

```
# su –
# cd /usr/local/caudium
# ./install
```

This script will allow you to give login information for the Configuration InterFace (CIF.), the web based configuration interface, and the port address for the CIF.. But if you want to start Caudium manually, you don't need to use install, just use the **start** script. This script will fork Caudium once and restart it automatically if it dies. A consequence is that if you kill **start**, the server will always be running but it will not restart if it dies.

There are many useful options to **start**. The first is `--help`. Here is a non–exhaustive list of options:

- `--once`: Do not fork Caudium and output debug to stdout (screen). If you hit **CTRL–C**, Caudium will be killed
- `-t`: Display all Pike calls
- `--gdb`: Run Caudium inside gdb, useful only for developers.
- `--with-threads`: Run Caudium with threads (run better on *BSD and Solaris)
- `--without-threads`: The opposite of `--with-threads`. It doesn't mean that the Pike scripts/modules aren't able to use threads. It merely means that the Caudium core server will not use threaded handler backend.
- `--with-keep-alive`: Enable keep–alive in HTTP. In the old days of the web, the HTTP protocol was simple but not efficient: one connection was made for each objects requested by a client. That means a web browser made 20 connections to the webserver if there was 19 images on a webpage and the HTML page itself. This result in a lot of overhead and response time delay. With keep alive, the server don't close the connection for each objects so the browser can request several objects with one HTTP connection and does not need to reconnect each time. As a result, the website seems to be faster for the client and the webserver can handle more users.

Currently, the keep−alive option doesn't work with CAMAS (I use CAMAS−1.1.7−DEV, Caudium 1.2RC1). Generally speaking, it is also not ready for production use. Here is a comment from Xavier Beaudouin:

> keep−alive is somewhat buggy on Caudium. My test shows that high number of connections on keep−alived Caudium show some random dropped returns. I do not recommend using keep−alive. If you'd like keep−alive a "black box" like redline seems the best solution... but expensive.

* `--config-dir=DIR`: Allows you to specify where your configuration files are, where "DIR" is the name of the directory holding the configuration files (typically `/usr/local/caudium/configurations/`). This is a very useful option. For example you can start several Caudium instances with different configurations by using different configuration directories. This is also useful if you put the configuration files in a non−standard directory: `/usr/local/caudium/server/start --config-dir=/home/david/etc/my_caudium_configuration/` For Apache users, this is the equivalent of the `-f` option but points to the directory that contains the files.

Finally, the most important thing is debug log files. These files are stored in `../logs/debug` (relative to `/usr/local/caudium/server` in our example). The current log file is named `default.1`. The log file from the last Caudium start is `default.2` and so on. If you didn't enable debug, these files are always used but contain very few messages.

The location of files may be different on your system if you are using a prepackaged version of the software.

4.3. Stopping from command line

There are two ways to stop Caudium:

* Kill Caudium with a `-9` signal. If you do this Caudium will restart. Actually it will be restarted by the start script.
* Kill Caudium (with kill(1)'s default TERM signal) . In this case, Caudium will stop cleanly. If you use Caudium 1.2 and later, you can use the **caudiumctl** command. A man page for **caudiumctl** should be available on your system.

In both cases Caudium will stop only after a few moments.

On Debian GNU/Linux, you can use the following commands to manipulate the server status:

```
# /etc/init.d/caudium stop
# /etc/init.d/caudium start
# /etc/init.d/caudium restart
```

4.4. Directory organization

* configurations:

Contains the config files used by the CIF.. These files use the XML syntax. You don't have to modify them by hand. If for some reason you do, make sure to do it when Caudium is stopped because they will get overwritten with the server's current configuration when the server is stopped. An alternate method is to edit the file, then send the HUP signal to the Caudium process. Another method is to use the CIF.: Go into `Action` −> `Maintenance` −> `Reload configurations from disk`. This will cause Caudium to reload its configuration from disk. As you may have guessed, each file has the name of the corresponding virtual server. One file however has a special name − `Global_Variables`. This file contains information about Caudium's global variables (See the Global Variables tab in the CIF.).

- logs:

  Contains useful information such as debug log, virtual server logs, start/stop status. See also Section 4.2.
- readme:

  It's all in the name.
- server:

  This directory contains the actual server.

  - base_server:

    Contains the server core Pike code: API, CIF., etc. Newbies should not modify these files.
  - bin:

    First binary code that is run when you use the start script. There are also some Pike scripts running as CGIs and without Caudium API.
  - caudium−images:

    This directory contains Caudium static images from the Caudium Group, such as the "powered by" series.
  - config_actions:

    These are files used in the `Action` tab on the CIF.. You may use some of these functions in your code.
  - etc:

    Caudium includes files used by the Caudium core and its modules. For example, it contains .html files for error messages. Two directories are important for a module developer:

    - include:

      Put your .h files here.
    - module:

      Put your module files here (.pmod). Pike module files are used to provide two−level functions/class/methods for your modules. When you modify any of these files, you must restart the server.
  - fonts:

    Contains some compatibility fonts with some older versions. The new directory is nfonts.

- languages:

  Some basic language translations for time/date.
- lib:

  Some of the C code used to speed Caudium up. This code is a set of dynamic libraries (.so).
- modules:

  Contains Caudium modules. These are all the modules you see when you click **Add Module** after you have selected one of your servers in the CIF.. Browsing this directory will be useful to understand Caudium.

  - 3rdparty:

    This special directory contains third party modules for Caudium. These may be useful but are provided without any warranty and will not be maintained by the Caudium Group but they may be by the individuals who wrote them.
- more_modules:

  Some modules that meet a specific need, are not up–to–date, or are humorous.
- nfonts:

  Fonts for graphical things like <gtext> and <gbutton>. You can have several types of properties like bold or italic for the same font name.
- Perl:

  Needed to run Perl scripts within Caudium. At the time of writing this document, Perl support is broken, and we will be pleased to get help on this issue.
- protocols:

  Contains protocol modules (for handling HTTP, FTP and the like).
- server_templates:

  The templates you can choose from when you create a new a virtual server.
- unfinished_modules:

  For modules in development.

You can also see `readme/README` relative to `/usr/local/caudium` in our example.

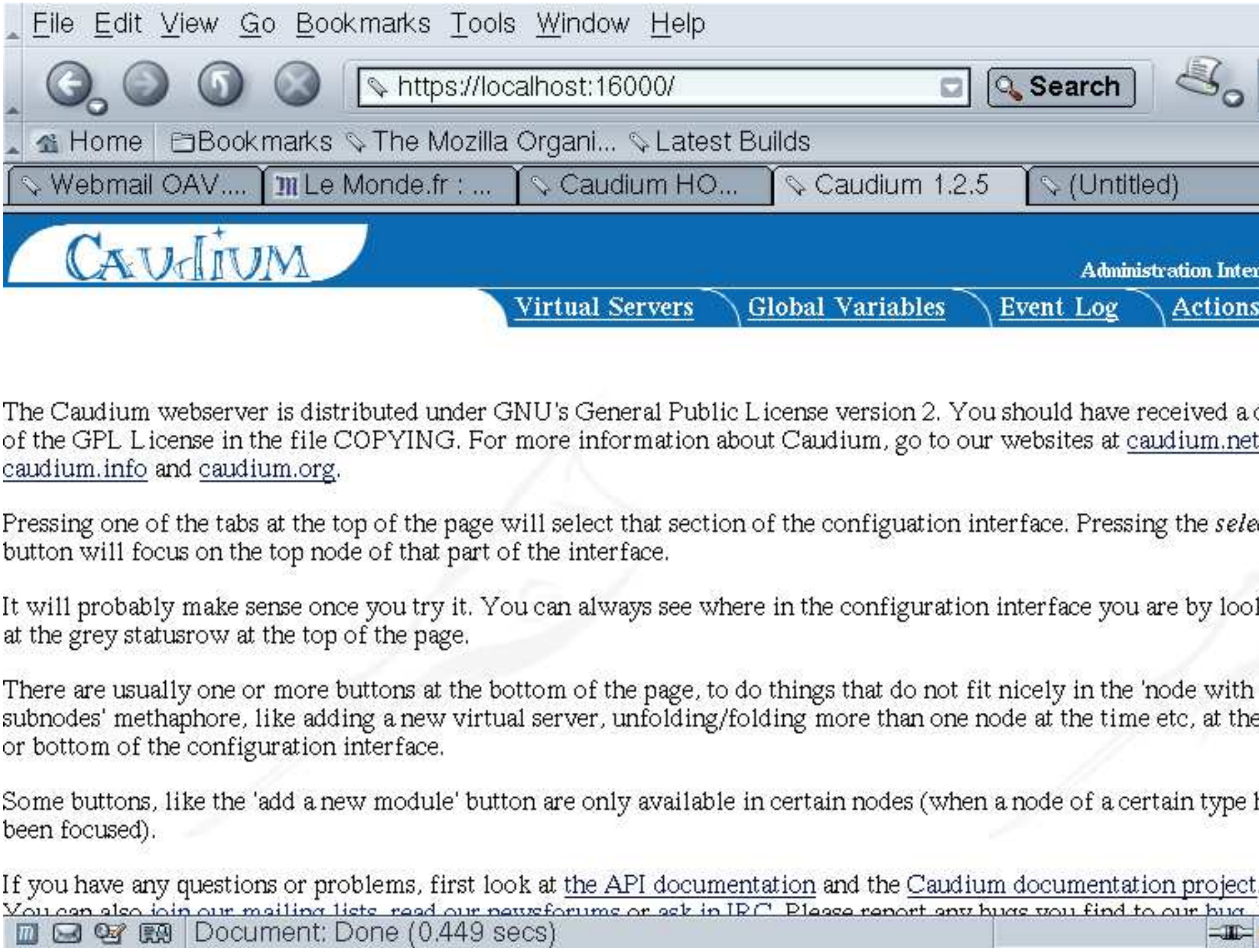
4.5. Upgrading Caudium

Upgrading Caudium is simple. You should install it as if for the first time. That is, get the new sources, unpack them, run `./configure; make; make install` and that's all. The new Caudium installation will detect your old Caudium and keep all your old installation including your config files. The config files will be compatible with the new version so you will never have to begin the configuration task a second time. The old server directory will be saved as server.old and so the new one will be put in the classical server directory.

If you use a package system, just use the upgrading features of your packaging system.

4.6. Configuration InterFace (CIF.)

The CIF. is where the administrator manages the server. When you first login it looks like this.



The first page you will see when you login on the CIF.

In the CIF, you'll find four tabs:

1. 

   The server(s) you have created in this Caudium installation. This is where you will tune each server configuration.
2. 

   The configuration variables that affect Caudium as a whole, including all virtual servers.
3. 

The log file output as a nice looking log. Note however, that neither errors nor startup events are displayed in the event log. This means developers should always look in the plain log file.

4. Actions

   Some actions you may take on you servers such as shutting down Caudium or generating an SSL certificate. This allows you to avoid some of the command line stuff.

As the CIF. is quite easy to cope with, I will try to describe some of the hidden things you may need to know about. One of them is the More options button. This button is very useful. Furthermore, some Basic options will not be available unless you turn it on. With this option, you will have more control over Global Variables and some modules in Virtual Servers.

One of these controls is the Reload module button that you will have when you are in a module. This button will allow developers to check the new code of their modules without restarting Caudium. You should also know that very few options in some modules will not be activated unless you reload the module. So always reload a module when you think an option has not been read by Caudium.

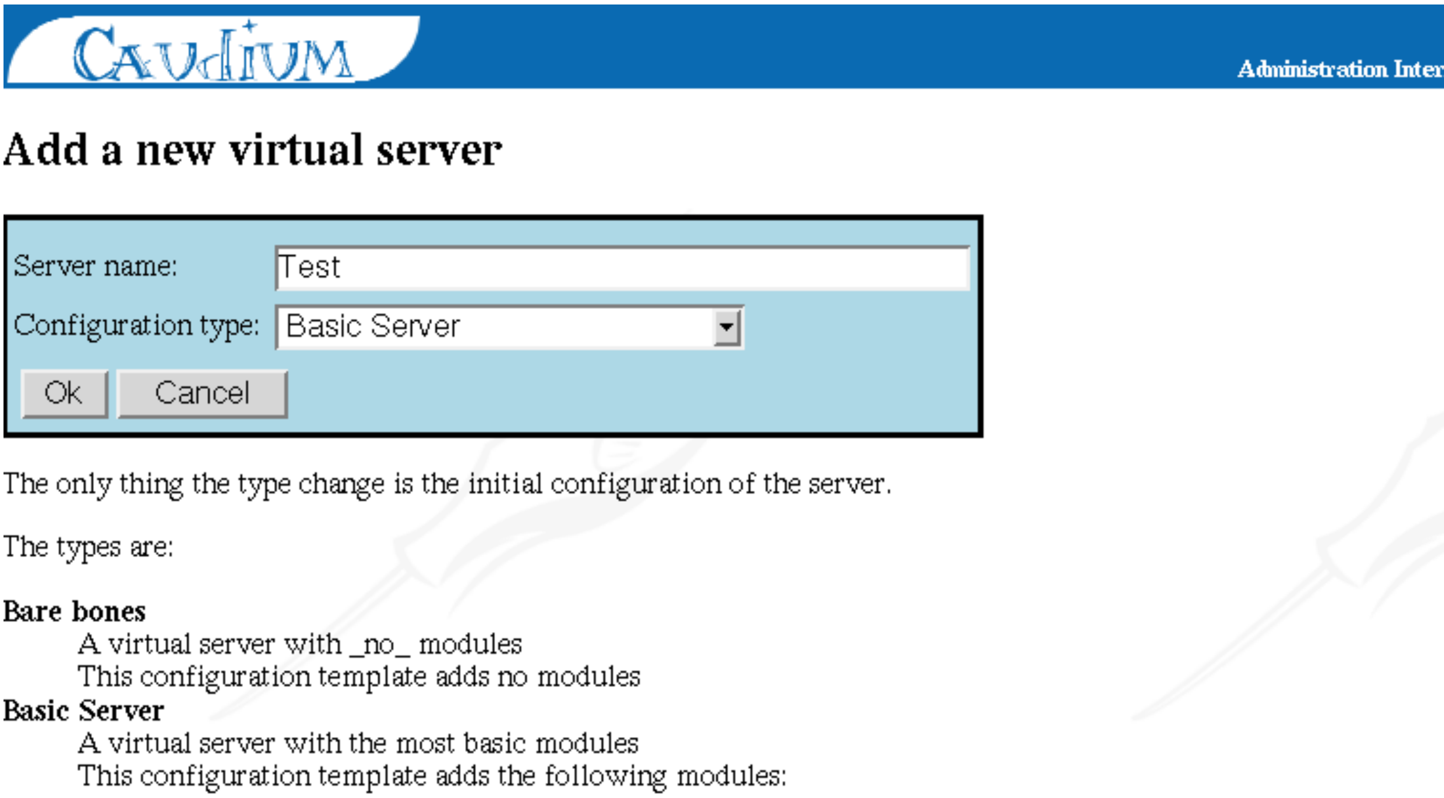
4.7. Adding your first server

In this section, I will give you a step–by–step tutorial on how to create your first site (virtual server) with Caudium. If you want to do something useful with Caudium, you have to create at least one virtual server. Without this first server, Caudium will not do anything. If you use your browser to access your server, you will only get a dialog box prompting for the CIF. login/password.

A virtual server allows you to have several servers running on the same port. For example you can have www.foo.com and www.foo.org running on the same port and machine. This is why it is called virtual server.

To create your first server, log into the web based CIF.. Click on the Virtual Servers tab, then the New virtual server button.

Here, you are prompted for the server name:



Type in an easily identifiable name. You also have to select the configuration type. Depending on the configuration you choose, your server will have a different set of modules. In other words, your server will have different capabilities. For your first server, choose `Basic server`, and click OK.

4.7.1. Server variables

Now you'll see the Virtual Servers page again. As you can see, you have two different folders to work on: Server variables (Figure 4–1) and File system (Figure 4–2).

Figure 4–1. Ports in the CIF

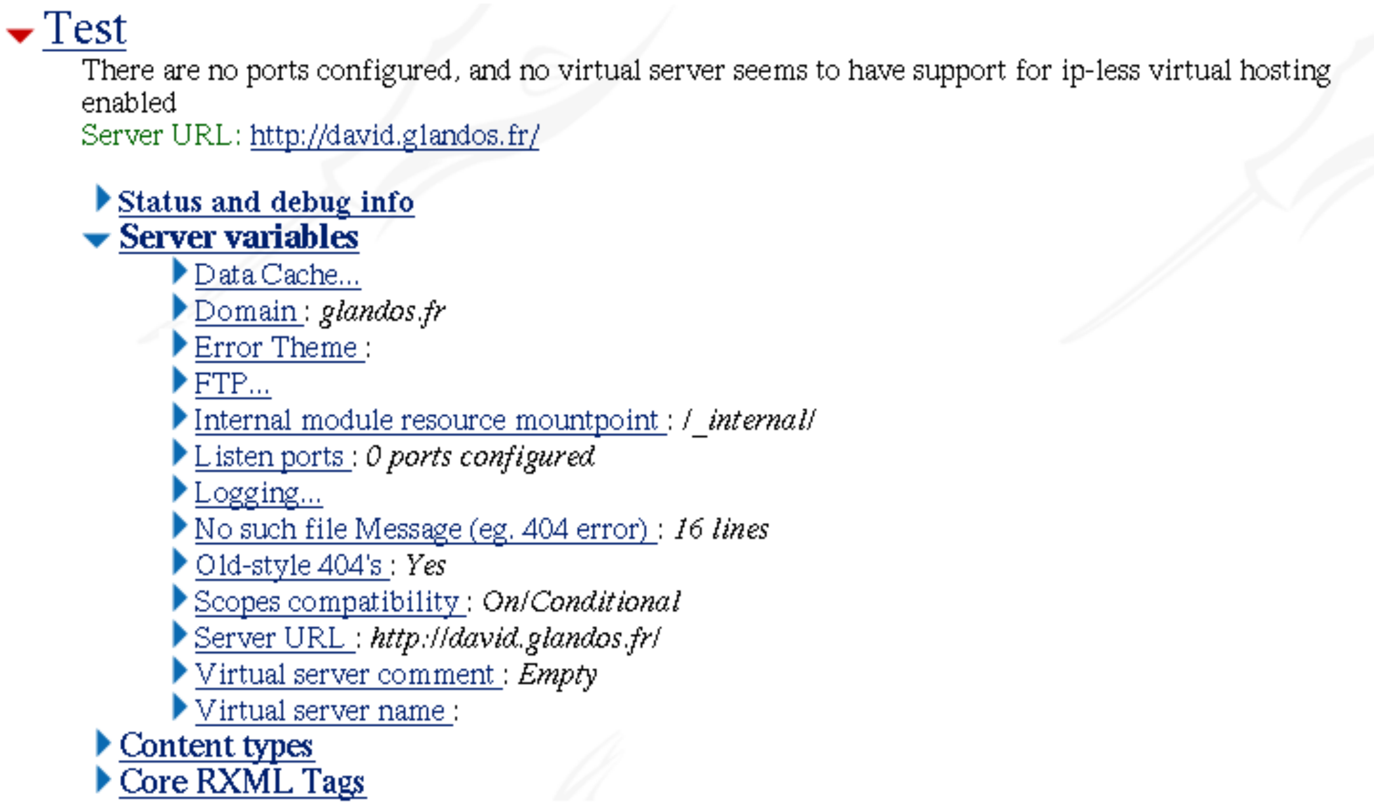
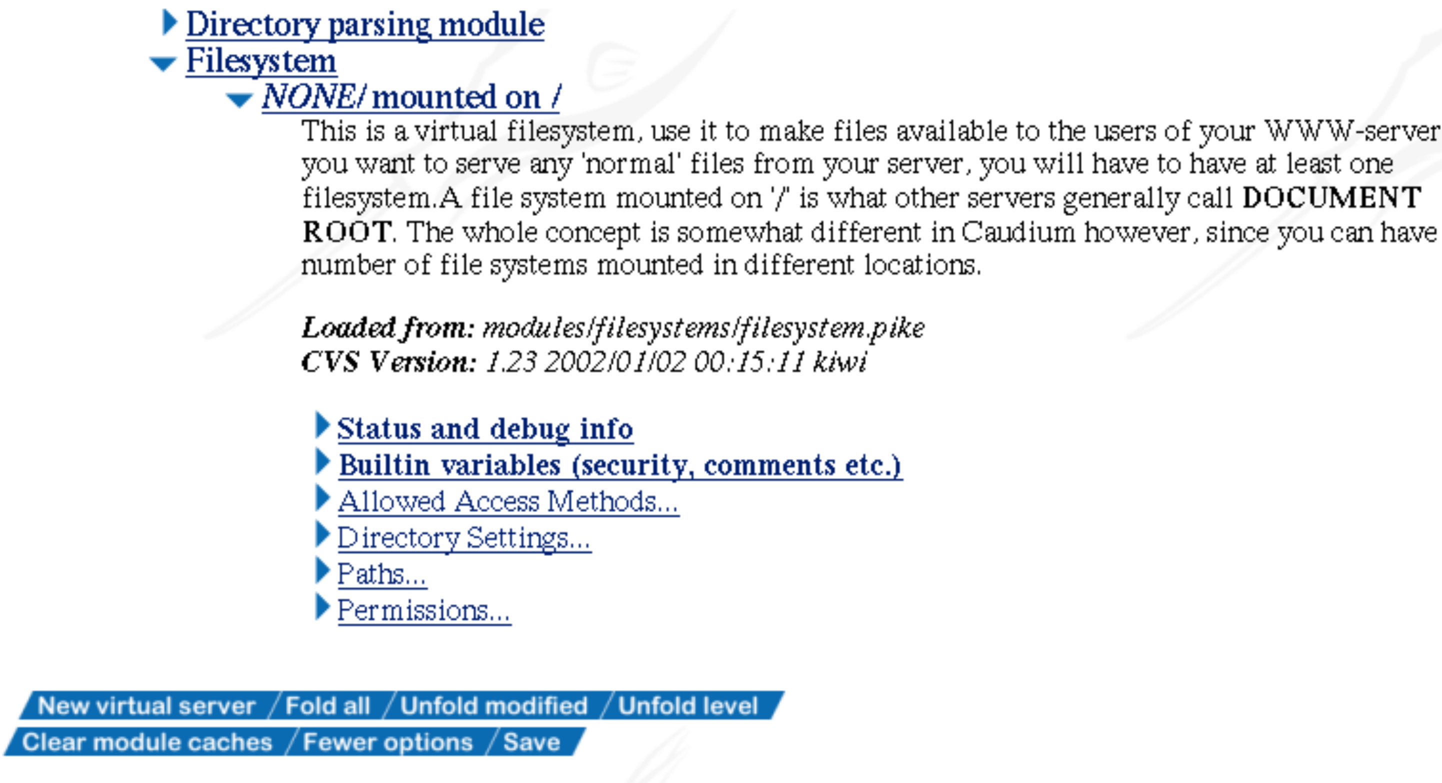


Figure 4–2. Filesystem in the CIF



Server variables contain the URL of your site, and the port on which it will be available. File system describes the files/directories containing the .html and other files you want Caudium to serve.

In server variables, go into Listen ports and choose Configure a new port. Use the default values, choose Use these values, and click Save . Now select the URL of your site, and select Continue. You can now go back to the page displaying all of your modules, that is, where you were before you had gone into Server Variables.

Under your server name, you'll see the status of your server. If it contains the word "Open" in blue everything is okay, and you can continue with configuring the file system.

If it contains the words "Not open" in red, there is a problem. You can go into the Event Log via the CIF. tab to investigate. If you have an error Failed to open socket on 0:80 (already bound ?), you may have another program or Caudium itself already running on this port. To fix the problem, identify the program which is using this port, and restart Caudium [1]. To restart Caudium, go into Action−>Shutdown−>Shutdown Caudium−>Restart Caudium. Wait a few seconds, and when prompted, select Virtual Servers. Now select your server, and you should have the word "Open" in blue.

In the CIF. you may wonder what are those two different protocols http and http2?

The difference is that http2 uses Caudium's internal memory cache while http is plain Pike http subsystem. So http2 is faster than http as you might expect. However, there are some issues on some specific sites.

4.7.2. Selecting file system

By selecting a file system, you tell Caudium which files it will send to people browsing your site. For those who know other web servers, please pay close attention to these explanations, because Caudium is quite different from other servers in this respect.

Caudium file systems use the Unix philosophy of mount point, rather than `c:`, `d:`, and so forth. The mount point concept allows you to put your files/directory under any URL you want without changing the files on your local file system. For example, assume you have the following local file system:

Example 4−1. Your user filesystem.

```
/home/customers/customer1
/home/customers/customer2
/home/customers/bigcustomer3
/home/friends/franck
/home/friends/bertrand
/home/friends/didier
```

And suppose your URL is http://www.iteam.org/.

With a default configuration, you would say that http://www.iteam.org points to /home/ so that you will have customer1 under http://www.iteam.org/customers/customer1, bertrand under http://www.iteam.org/friends/bertrand and bigcustomer3 under http://www.iteam.org/customers/bigcustomer3.

But bigcustomer3 gives you a lot of money and he asks you for an URL such as http://www.iteam.org/bigcustomer3. However, he doesn't want to be moved from /home/customers/bigcustomer3 because of his FTP client's configuration. Moreover, you can't move the other accounts. With the mount point philosophy, you just have to create another mount point saying that /home/customers/bigcustomer3 is mounted on /bigcustomer3 so that when someone uses http://www.iteam.org/bigcustomer3, Caudium will serve them files from /home/customers/bigcustomer3.

Now let's return to our setup. Go into the `File System` module, and select `NONE/ mounted on /` –> `Path` –> `Search Path`. Here you will decide which of your directories will be available when someone hits the root of your server. Write, for example, **`/home`**.

You can now launch your favorite browser to the URL of your site and enjoy.

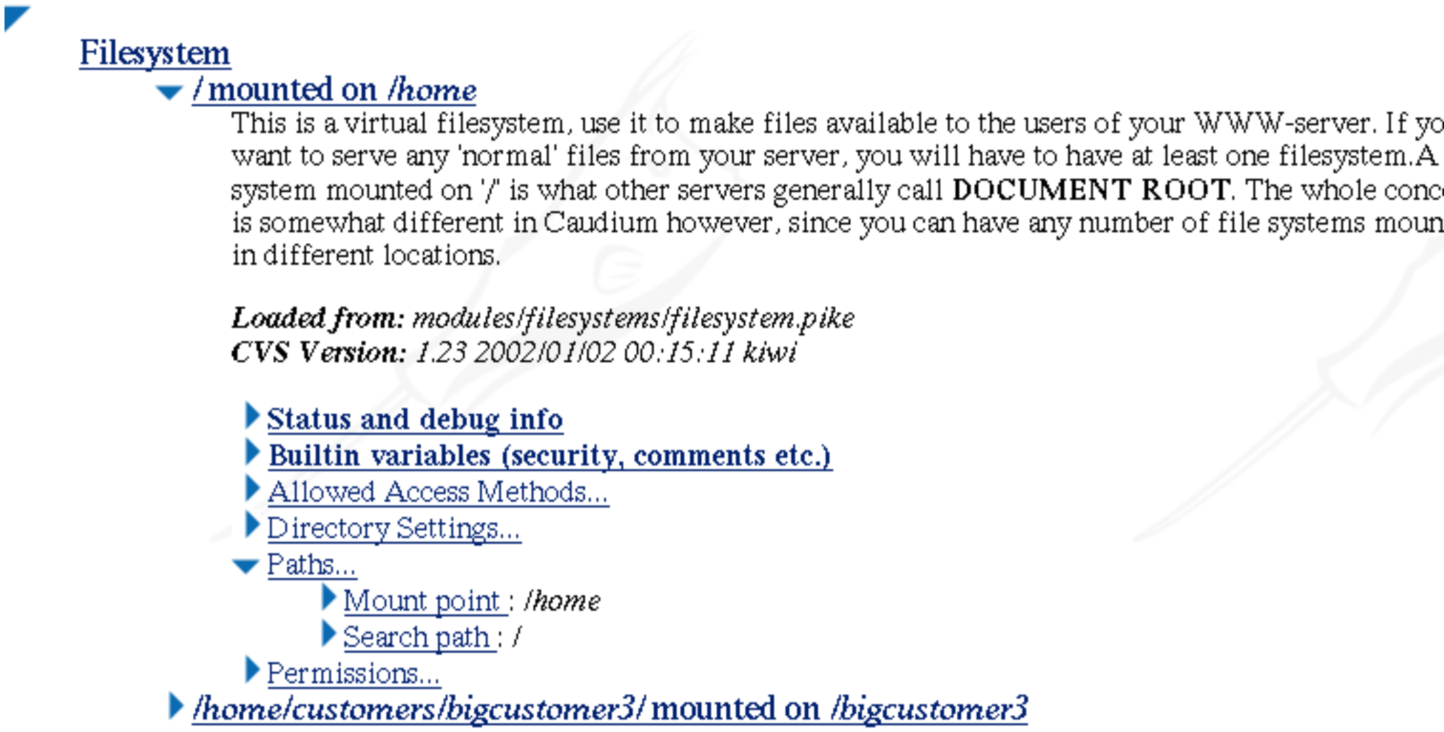
If you don't want visitors to get a listing of your files for security reasons, you can disable listing by setting `Directory Settings` –> `Enable directory listing per default` to `No`. Next you have to hit the `More options` button, and then reload the module. Don't forget to click `Save`. It is possible to put one of two "magic files" in any directory to make it browsable/not browsable despite the setting in the CIF. for that particular file system:

```
.www_browsable – the directory will always be browsable
.www_not_browsable – the directory will never be browsable
```

Finally, to allow bigcustomer3 to get to his URL, go to `File System` –> `Copy Module`. Next, go to `Path` –> `Search Path` and type **`/home/customers/bigcustomer3`**, then type **`/bigcustomer3`** in `Mount point`.

The Figure 4–3 shows the output you should have.

Figure 4–3. Example of output



If you created these files/directories after you point Caudium to your site, it is safe to go in the `Actions` tab, then to `Cache` –> `Cache status` –> `Flush caches`.

4.7.3. Creating a virtual server

A web server is usually running on a single port and on a single IP address. So how can someone have different sites on this single port and single IP? The solution is to tell Caudium that the server is different based on the URL. This is the task of the Virtual Hosting module. As this module is not present in the Generic template we use, you will have to add it. Just use the `Add module` button, and click the image named "Virtual Host Matcher".

You are now able to do redirection with the help of regular expressions. If you don't know what regular expressions ("regexp" for short) are, check the man page for regexp, see Pike/Perl manual, or maybe buy a book on regular expressions. To write these redirections based on the URL, go into `Regular expression rewrite rules` and add the following rule:

Example 4–2. A simple virtual hosting regular expression.

```
www\.virtualhost\.com    www.virtualhost.com
```

Example 4–3. A better and quicker regular expression.

```
^www\.virtualhost\.com$|^virtualhost\.com$ www.virtualhost.com
```

Save and type this command on your command line in order to get your browser resolving www.virtualhost.com:

```
# echo "127.0.0.1   www.virtualhost.com" >> /etc/hosts.
```

You now have to add another virtual server with the button `New Virtual Server` at the root of the virtual server tab. Put **`Virtual host`** as the server name and choose the `Generic server` configuration type. Go into `Server variables` −> `Server URL` and type **`http://www.virtualhost.com/`**. You should see:

This server is handled by the port in my first virtual server. Server URL: http://www.virtualhost.com/

Change the file system root and /tmp in `File system` −> `NONE/ mounted on /` −> `Paths` −> `Search path`.

You can now point your favorite browser to http://localhost/ and http://www.virtualhost.com/ and see the result.

If you have an error telling you www.virtualhost.com is unknown check your host file.

For more information about virtual hosting, see the Virtual−Web Mini−HOWTO available at the LDP or in `/usr/share/doc/HOWTO` or `/usr/doc/HOWTO` under Debian GNU/Linux.

Chapter 5. Customizing your server

5.1. How to run Caudium as a non−privileged user; How to secure Caudium

Web servers are usually publicly accessible and represent your company, group or entity so there are chances you want to strengthen the security of this service.

As I already mentioned Caudium has a good security for public access behind mostly written in a script language. However Caudium runs as root by default. In the case a non−authorized user gains access to Caudium's process, he might gain root privileges. Consequently,a lot of web servers run as another user with minimal privileges. Doing this may require some work, as you will have to change the owner of all the files Caudium needs access to, so I give step−by−step instructions how to change those permissions:

1. Find a good user name. This user name should be a normal user with the least privileges. Lots of distributions already have a special account for this. Common names include "www", "www−data", "httpd", "nobody" (Caudium on Debian GNU/Linux runs as www−data:www−data by default). We don't recommend "nobody" though; to quote Theo de Raadt:

   > The user "nobody" has historically been doing too much. If you could break into the user "nobody", you could cause great damage.
2. Change the owner of the files which Caudium needs to write to. These include:

   - Caudium internal log file (default.*).
   - Per virtual server log file.
   - All caches.
   - The configurations files (they are written by the CIF.).

   On a Caudium source install the following command should do the job:

   ```
   # chown −R  www-data.www-data logs/ var/
   argument_cache/ bgcache/ configurations/ server/*.pem server
   ```

   Here is the result:

   ```
   $ ls −l
   total 32
   drwxr−sr−x    6 www-data www-data     4096 Feb 13 23:17 argument_cache
   drwxr−sr−x    2 www-data www-data     4096 Feb 19 09:27 bgcache
   drwxr−sr−x    2 www-data www-data     4096 Mar  4 22:28 configurations
   drwxr−sr−x    4 root     staff        4096 Feb 13 23:16 local
   drwxr−sr−x    7 www-data www-data     4096 Mar  3 11:50 logs
   drwxr−sr−x    2 root     staff        4096 Feb 13 23:16 readme
   drwxr−sr−x   19 www-data www-data     4096 Feb 19 20:13 server
   drwxr−sr−x    2 www-data www-data     4096 Mar  3 19:28 var

   $ id www-data
   uid=33(www-data) gid=33(www-data) groups=33(www-data)
   ```

   If users are allowed to log on the server, you might also change the permissions of the logs directory.

If you have a Caudium specific distribution for your system (such as Debian GNU/Linux) check manually.
3. Don't forget to change the permissions of any script/directory you made and for which Caudium needs to write to in your public filesystem.
4. Log into the CIF., go in the main `Global variables` tab, then in `Change uid and gid` type the uid:gid data you choose. We typed **`33:33`** in our example. You can also type a login name and group name: **`www-data:www-data`**. You can also enable the `Change uid and gid permanently` option but be sure to read the documentation first.

I will now speak about general security measures you can take if you are very strict about security.

1. Don't allow users to execute scripts that are part of the server.

   As Caudium is a single process server, it is possible to stop it, restart it, access it, etc. with a user script. This include pike scripts, pike tag, and PHP modules for Caudium.

   If you do want to let your users run scripts, you can always use CGI, or better uniscript (in this case it will be transparent to the user), in order to run a script in a separate process using the fork(2) system call. This will decrease the performance of Caudium but the security has a price, and it is up to you to decide how much you want to pay.

   Uniscript is a CGI−like wrapper. It will execute programs as if they were CGI scripts but unlike CGI, it does not require you to put these programs under a specific directory like /cgi−bin/. For example each user can have his or her CGI script in his or her directory. Moreover Caudium can execute them with the uid of the owner.
2. Don't use anything you don't need. Remove any modules you don't need in your virtual server.
3. Physically restrict access to the CIF.. Don't access it from the Internet if possible. Few people know this, but it is now possible to see SSL connections in clear text with a man−in−the−middle attack. The dsniff software contains all the tools and explanation for this.
4. Turn off these options:

   ♦ `Global Variables` −> `show_internals`.
   ♦ `Global Variables` −> `Version numbers` −> `Show Caudium Version Number`.
   ♦ `Global Variables` −> `Version numbers` −> `Show Pike Version Number`.

   Turn off any debug options specific to a module. These options are for developers, and they don't have security in mind when they debug output.

   > Actually, this is security through obscurity and doesn't increase the security of the server.
   >
   > −−Grendel
5. Output Caudium's log files to a separate partition. `/var` is a good choice for that purpose.
6. Check the Caudium web site for patches.
7. If your job relies on your web server security, check the Caudium source.

5.2. How to benchmark a web server

First, benchmarking a web server is not an easy thing. To benchmark a web server the time it will take to give a page is not important: you don't care if a user can have his page in 0.1 ms or in 0.05 ms as nobody can have such delays on the Internet.

What is important is the average time it will take when you have a maximum number of users on your site simultaneously. Another important thing is how much more time it will take when there are 2 times more users: a server that take 2 times more for 2 times more users is better than another that take 4 times more for the same amount of users. If you run more than a web server on your computer, you will also want to look at the load average and CPU time of your system. Here is a typical output of the command **uptime**:

```
22:39:49 up 2:22, 5 users, load average: 0.01, 0.01, 0.00
```

And an extract from the top(1) man page:

" The load averages are the average number of process ready to run during the last 1, 5 and 15 minutes "

So the lower your load average is, the better for the other programs on your machine.

Now comes the next problem: how can you stress your web server with a maximum number of connections when your client (the machine making the request) will usually not be able to cope with the server and with the number of users you have.

To do this, increase the number of sockets you can have on your system. Under some systems it is 1024, which is too low, see Section 5.3 for more information. The next thing to do is to have a good client program written with threads and non−blocking sockets. If you use a multi−fork program on a single client, it will never cope with any web server. It is also good to have several clients stressing the server together.

Last, if you want to compare two web servers, be sure that they are on the same hardware, OS, and network. The same holds for the client(s).

5.3. How to tune your system for best Caudium performances

5.3.1. Linux

Until a Linux guru has time to make some nice documentation, here is something that came from PureFtpd software :

```
o Increase your system max descriptors numbers :

 # echo 60000 > /proc/sys/fs/file-max
 # echo 180000 > /proc/sys/fs/inode-max
 # ulimit -n 60000
 o mount your filesystems with the "noatime" option
 o make sure your disks holding the logs are "fast enough"

 o You can tweak a bit your TCP/IP stack :
 # echo 0 > /proc/sys/net/ipv4/tcp_syncookies
 # echo 0 > /proc/sys/net/ipv4/tcp_ecn
 # echo 0 > /proc/sys/net/ipv4/tcp_timestamps
 # echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
```

Finally don't forget to compile Pike with `--with-max-fd=60000` (already done in Debian packages).

Warning from Caudium people

- Personally deactivate window scaling seems to be a bad idea.
- This needs some touches. 2.2 and 2.4 options are mixed above very badly. Also, turning off ECN might do you more harm than gain.

Also see LinuxPerf, and LinuxPerf kernel tuning section.

5.3.2. FreeBSD

Here are the optimizations you can try on your servers. They are provided without any warranty.

All this is for FreeBSD 4.2 or more recent.

First, check to see if your filesystems use `Soft Updates`:

```
# tunefs -p /dev/da0s1a
tunefs: soft updates:  (-n)                                 disabled
tunefs: maximum contiguous block count: (-a)                15
tunefs: rotational delay between contiguous blocks: (-d)    0 ms
tunefs: maximum blocks per file in a cylinder group: (-e)   2048
tunefs: average file size: (-f)                             16384
tunefs: average number of files in a directory: (-s)        64
tunefs: minimum percentage of free space: (-m)              8%
tunefs: optimization preference: (-o)                       time
```

If `soft updates` are set to `disabled` it may be a good idea to enable them. We do not recommend you enable them on "/" filesystem on servers machines, there are issues and in general this is not recommended by the FreeBSD team. If you can unmount the others filesystems do this and remount them later:

```
# tunefs -n enable /dev/"whatever"
```

If you cannot unmount the others filesystem, drop into single mode (do a **shutdown now** or a **boot -s**) and then type:

```
# tunefs -n enable "filesystem"
```

I suggest /usr or /var. In the fstab you can add **`,async`** to the options of all filesystems. As for soft updates we do not recommend that on server machines for the root (/) filesystem.

In `/boot/loader.conf` add the following:

```
kern.ipc.maxsockets="5000"
kern.ipc.nmbclusters="65536"
```

If you have ATA (IDE / UltraDMA) disks you can add also in `/boot/loader.conf`:

```
hw.ata.wc="1"
```

Then in `/etc/sysctl.conf` you can add:

```
vfs.vmiodirenable=1
kern.ipc.maxsockbuf=2097152
kern.ipc.somaxconn=8192
kern.ipc.maxsockets=16424
kern.maxfiles=65536
kern.maxfilesperproc=32768
net.inet.tcp.rfc1323=1
net.inet.tcp.delayed_ack=0
net.inet.tcp.sendspace=65535
net.inet.tcp.recvspace=65535
net.inet.udp.recvspace=65535
net.inet.udp.maxdgram=57344
net.local.stream.recvspace=65535
net.local.stream.sendspace=65535
```

And then reboot (or for `/etc/sysctl.conf` use the sysctl(8) tool to setup this by hand.

Another way to have good performance is to make a custom kernel, with the minimum of drivers and processor support. A maxuser size = size of memory (for example you have 512M of RAM, then set the maxusers value in your kernel configuration variable to 512). Add Posix 1003.1b real time extensions to the kernel with:

```
options         P1003_1B                #Posix P1003_1B real-time extensions
options         _KPOSIX_PRIORITY_SCHEDULING
```

Enable the SMP options (only for FreeBSD 4.x, the FreeBSD 5.0 is now automatic).

For French users, check these URLs :

- http://gcu−squad.org/?viewtip+&tip_id=59.
- http://gcu−squad.org/?viewtip+&tip_id=54.

5.3.3. Solaris 2.x

Most of the optimization values are in `/etc/system` file. This file is read by the kernel when it is loading. Please notice that the following optimization is focused on Solaris 8 machines with at least 256M of RAM.

```
* A comment is started by a star '*'
*
* 1 maxusers per mega of ram. This machine has 512M then maxusers = 512
set maxusers=512
set hires_tick=1
set rlim_fd_max=10000
set rlim_fd_cur=4096
*set tcp:tcp_conn_hash_size=8192
* Don't setup this if you don't have more than 256M of RAM
set bufhwm=48000

* Folling are used to delay page reaping with databases.
set fastscan = 32000
set slowscan = 200
set maxpgio =  280
set lotsfree = 1024
set desfree = 512
set minfree = 128
set autoup = 280
```

```
*  Hash buffer sizes during Specweb testing
set tcp:tcp_conn_hash_size = 2097152

* Nic Interface
set hme:hme_adv_100fdx_cap=1
set hme:hme_adv_100hdx_cap=0
set hme:hme_adv_10fdx_cap=0
set hme:hme_adv_10hdx_cap=0
set hme:hme_adv_autoneg_cap=0

* To prevent buffer overflow
set noexec_user_stack=1
set noexec_user_stack_log=1
```

Please notice that Solaris needs fast disks. If you have IDE / UDMA disks, double check they are recent. For example the disks in Ultra 5 / 10 machines are snail slow, and slow down the whole machine when you make read write operation. Please consider getting faster disks.

Also, if you use software raid (Solaris Disk Suite) you will have optimal performance if you have more than one SCSI controller in the machine.

Another good read is the well known document from Adrian Cockroft about tuning Solaris. Sunhelp.org has also a good section about tuning. Finally, you can read a network guide at http://www.sean.de/Solaris/tune.html.

5.4. How to use your own fonts

Use the xdumpfont program in the tool directory of Caudium's sources. This program will convert your X fonts to Caudium fonts.

If your Pike supports TrueType Fonts, (type **`pike --features`** and check if `Image.TTF` is available) you can import TTF font by copying them into `/usr/local/caudium/server/fonts/ttf`.

5.5. How to get UltraLog working

See the Caudium documentation project at http://caudium.info/, and http://daviesinc.com/modules/docs/ultralog.rxml.

Chapter 6. Developing with Caudium

6.1. Your first RXML file

You can try this code. This is a basic one with only four tags/containers. You should go to http://caudium.info/ and download the Roxen 1.3 documentation.

Example 6−1. Some simple RXML tags.

```
<html>
  <comment>You have to put bgcolor for gtext to work properly</comment>
  <body bgcolor="#FFFFFF">
    <h1>Basic RXML examples</h1>
    <table border="1">
      <tr>
        <td>This is a list of all RXML tags</td>
        <td><list-tags></td>
      </tr>
    </table>
    <table border="1">
      <tr>
        <td>Gtext render text as graphic</td>
        <td><gtext scale=0.5>This is a gif/png graphic</gtext></td>
      </tr>
      <tr>
        <td>Last modification of this page</td>
        <td><modified></td>
      </tr>
    </table>
  </body>
</html>
```

6.2. The Pike tag

For a complete tutorial and reference manual on Pike, see http://pike.oav.net/.

The Pike tag allows you to easily insert Pike code into your HTML page à la PHP. This is a good way of learning Pike if you have a PHP background or if you want to do things very easily and don't worry about perfect results.

To do this, you have to load a module in your server. Just select `Load module` in the CIF. and click the "Pike tag" image. Then hit save and create an .html file where your public web files are. Since everybody tells me that PHP is easy, which is why it's so popular, I took the PHP examples and converted them to Pike. Here is the result:

Example 6−2. The PHP documentation as a Pike tag.

```
<html>
  <body bgcolor="#FFFFFF">
    <h2>Escaping from HTML</h2>
```

```
    <p>
      There is one way of escaping from HTML and entering "Pike code mode"
      <br />
      <Pike>
        output("This is the simplest, and SGML processing instruction");
      </Pike>
    </p>
    <h2>Instruction separation</h2>
    <p>
      Instructions are separated the same as in C or Perl:
      terminate each statement with a semicolon.
      The closing tag (container, in fact) also implies the end of the statement,
      so the following are equivalent:
      <br />
      <Pike>
        output ("This is a test");
      </Pike>
      <br />
      <Pike> output ("This is a test"); </Pike>
    </p>
    <h2>Comments</h2>
    <p>
      Pike supports C, C++ but not Unix shell-style comments. For example:
      <Pike>
        string tests = "This is a test<br />";  // this is a one-line C++ style comment
        /* This is a multi line comment
         yet another line of comment */
        tests += "Yet another test";
        return tests;
      </Pike>
    </p>
    <h2>Melding RXML and Pike</h2>
    You will never see this in PHP.
    [2]
    This will create a .gif image of 1-2-...-255.
    This took 0.4s on my Duron 750 the first time and 0.1s after.
    <p>
      <gtext scale=0.5>
        <Pike>
          string output = "";
          for(int I = 1; I < 255; I++)
            output += I + "-";
          return output;
        </Pike>
      </gtext>
    </p>
  </body>
</html>
```

The problem with using this is that you will soon see it is not powerful:

- The code will become an ugly mixture of data and code even if you use an object orientation method. The more your project grows, the less you will be able to produce something useful.
- The URL will require plenty of those "&" symbols and you will have a lot of files without real organization.
- The performance will not be the best.
- This can be considered as a security risk for servers that are opened to unknown users (eg. public web servers).

The next step is to write your first Pike script.

6.3. Your first Pike script

A Pike script is quite like a Perl script. It is executed when the user tries to access it. So a Pike script is usually where your public web files are. This is a good choice if you already have a Perl background and want to try Pike.

You have two choices when doing this. You can execute Pike as a CGI script or internally within the server. If you don't know what CGI is, look up the Apache−Overview−HOWTO at http://www.tldp.org/.

Here, we will run Pike scripts internally within Caudium. To achieve this, you have to load another module in your server by selecting `Load module` in the CIF.. You now have the list of all modules available in Caudium. As you see, there are a lot of modules and reading this page should give you some ideas for future development. To select the Pike script module, just click on the image named "Pike script support" if you use a graphical browser.

You can now create a .pike file containing, for example,

Example 6−3. A basic Pike script.

```
// you have to inherit caudiumlib to have some basic things
// like the id object and response mapping
inherit "caudiumlib";

// the same as the main
// if you modify this script and you see that Caudium don't take your
// modification into account, reload the Pike script support module
// This is because Caudium uses a cache for performance reasons
string parse(object id)
{
  string html = "<html><body>The id object contain some "
  "very useful information<br />";
  html += sprintf("The id−>variables contain a list of "
  "arguments given to this script %O\n", id−>variables);
  return html;
}
```

Pike scripts are usually used for little internal development. Pike scripts can be very useful in this case because you can create something with very little lines. Here is an example of such a script:

Example 6−4. A real world script.

```
/* Here is a Pike script (not a Caudium module).
   This script is less than 20 lines (comments
   and blank lines excluded) and will randomly
   return one file to the web browser from a list of files.
   This script was kindly provided by Xavier Beaudouin */

// first we need to inherit from caudiumlib in order to get
// the parse, http_redirect functions and id object
```

```
// recognized.
inherit "caudiumlib";

// we declare an array of files
array (string)files;

// an ASCII text containing the name of a file
// on the real filesystem.
// Each file name in this file will be
// randomly return (the files name have to be on
// a separate line).
#define FILELIST "/list"

#define BASEDIR "/thepath2yourfiles/"

// this function is the constructor, it will be loaded first
void create () {
 // the array of strings 'files' will contain
 // all the files we serve provided the file
 // FILELIST list each file name on one line.
 files = Stdio.read_bytes(FILELIST)/"\n";
}

// if no_reload return 1, Caudium will cache the
// result of this script for maximum performances
// and will not execute it a second time.
// As a result, If you give the argument
// ?reload=1 to your script, Caudium will
// reload it.
// This is useful to use cache for average
// content delivery unless you are doing
// developpement
int no_reload(object id)
{
 if(!id->variables->reload)
   return 1;
 return 0;
}

// As this is a simple pike script (CGI like), this function
// will be called by Caudium and should return a string that
// will be display to the client's browser.
// It can also return a mapping containing all the HTTP response
// (headers + text)
mapping parse(object id)
{
 // We randomly return one of the file we list in the FILELIST file
 // (relative to BASEDIR directory).
 // http_redirect will send a HTTP 301 header telling the browser
 // where to get randomly selected file.
 return http_redirect(BASEDIR + files[random(sizeof(files))],id);
}
```

But you can also create some powerful scripts:

Example 6–5. A script for the power user.

```
inherit "caudiumlib";
```

```
string|mapping|object parse( object id )
{
  id->my_fd->write(id->clientprot + " 200 Ok\r\n");
  id->my_fd->write("Server: Caudium !\r\n");
  id->my_fd->write("Expires: 0\r\n");
  id->my_fd->write("Content-Type: text/html\r\n");
  id->my_fd->write("pragma: no-cache\r\n\r\n");
  id->my_fd->set_id( ({ id->my_fd }) );
  id->my_fd->set_nonblocking(0,send_data);
  return http_pipe_in_progress();
}

void send_data(array (object) id)
{

  id[0]->write("<pre>");
  id[0]->write("test......................\n");
  id[0]->write("test......................\n");
  id[0]->write("test......................\n");
  id[0]->write("sleep for 10 sec\n");
  sleep(10);
  id[0]->write("Done</pre>");
  id[0]->close();
  destruct(id[0]);
}
```

This example uses non−blocking sockets. my_fd is the file descriptor of the HTTP socket. Here we change the type of the HTTP socket from blocking sockets (default type) to non−blocking sockets. Non−blocking sockets are sockets that won't block the program waiting for data. Instead, a `read` and `write` function (the so−called callback functions) will be called automatically when there is some data to read or write to the HTTP socket. Moreover, we return here a special function, `http_pipe_in_progress`. This is because as the HTTP socket is set to non−blocking, Caudium won't be able to wait for processing the HTTP stuff like headers and so on. So we have to tell it not to wait for us and send a `http_pipe_in_progress`.

This mechanism is very useful when you have to do some communication with slow sockets on a single process server (multi−threaded one). In the case of a single process, when you wait for a socket it is all the server, which will wait. So all your users will be stalled. With non blocking sockets there is no problem anymore; the server won't wait for each socket. Example of such code includes CAMAS IMAP/NNTP clients. If you don't understand, don't worry, you usually don't have to understand these mechanisms.

However, the Pike script allows you to write some complex code it is not well suited for big projects. If this is the case, read the next paragraph and enjoy.

> The Caudium API is available at http://caudium.net/. You should also read the Roxen 1.3 PDF available at http://caudium.info/. Pike scripts are blocking, and allow your users to run scripts with the same privilege as the server. Blocking means that the server will be stalled if a socket from a pike script is stalled (usually waiting for something). This applies even if you use non−blocking sockets in your script. You don't have this problem with modules.

6.4. Your first module

With a custom module you can do all sorts of things:

- You can create a professional quality administration center very easily.
- You don't need any more "&" symbols in the URL.
- You can also use `per user variables`, also known as `session variables`.
- You'll get better performance since the module is part of the server.
- You can separate big projects into different modules, and do calls between different modules. This way, your project is not a big complex of messy code, but a set of simple, easy to extend code modules.
- You can separate data from code by using tags and containers. This also allows you to delegate the appearance to your webmaster, and lets you focus on the important code.
- You can easily share your code with the Caudium community. If your code is good and useful, it can become part of the Caudium distribution. This way more people will test it, you will have more feedback, and some people may help you with your project, and may even maintain it.

There are different types of modules, for example:

- Location:

  This is the most common module, your code is called when the user hits the URL you specify in the mount point.
- Parser:

  Your code is called when Caudium parse a file containing the tags and/or containers you define.
- Authentication:

  Used to authenticate users with, for example, LDAP, shadow, or SQL.
- Directory:

  For indexing files in a directory.
- First module:

  Module that is called just after the authentication module, thus letting you handle the whole request before normal processing.

There are other module types. For a complete reference see the Roxen 1.3 Programmer's Guide at http://caudium.info/.

For an example on how to write a container, see `fnord.pike` in `/Caudium/sources`. Because the location module is a must, here is another example:

Example 6−6. A sample module.

```
// It is intended to show a simple example of a location module.

// This module output the result of the who(1) command. It is not meant to
// be really useful since it would be better to do a <who /> tag thus
// having data in the HTML files and code here
```

```
// This variable is shown in the configuration interface as the
// version of the module.
string CVS_version = "$Id";

// Tell Caudium that this module is 'thread safe', that is, there is no
// request-specific data in global variables.
int thread_safe=1;

#include <module.h>
inherit "module";
// for the http_string_answer API
inherit "caudiumlib";

// Documentation:

constant module_type = MODULE_LOCATION;
constant module_name = "Who";
constant module_doc  = "Send the result of the who(1) command ";
constant module_unique = 1;

// The constructor of this module.
// This function is called each time you/Caudium load the module
void create()
{
  defvar("location", "/who", "Mount point", TYPE_LOCATION,
  "The mount point of this module");
  /* each string have to be on a single
   line, don't do: "The mount point of
   this module".
   You can however do "The mount point of "
   "this module";
   */
  defvar("path2who", "/usr/bin/who",
  "Path to the who command", TYPE_FILE);
  defvar("options2who", "-a",
  "Options given to who", TYPE_STRING);
  defvar("codebeforewho", "<html><body><p>",
  "The code to output before who", TYPE_STRING);
  defvar("codeafterwho", "</p></body></html>", "The code to output after who",
  TYPE_STRING);
}

// This function is called when a user access mount point
// path is the path to the URL he used
// id contains Caudium global variables such as browser name,...
mixed find_file(string path, object id)
{
  // get the contents of the CIF. variables path2who and options2who
  // and put a single space between it.
  string command = QUERY(path2who)+" "+QUERY(options2who);
  // this will write the result of command to the debug log file
  // very useful for debug
  write(sprintf("command=%s\n", command));
  string result = Process.popen(command);
  // replacing \n by \n<br /> for better output
  result = replace(result, "\n","\n<br />");
  return http_string_answer(QUERY(codebeforewho)+result+QUERY(codeafterwho));
}
```

Put this code in `../local/modules/who.pike` relative to `/usr/local/caudium/server` in our example. Log into the CIF., if it is not the case and go into the main `Action` tab –> `Cache` –> `Flush caches`. Check the `Module cache` check the box and press `Next`, then `OK`.

Come back to the main `Virtual servers` tab and choose one of your servers. Do `Add module` and select the `who` module. If you don't have the who module, check your events log.

You don't need to compile to have a working module. You don't even need to restart the web server. When you develop a module and change the code every 30 seconds, you just have to push the `Reload` button to get the changes. It takes about one second and if there was a compilation error the old copy remains for users.

6.5. How to use a backtrace

A backtrace is text that will show you where your program come before the error. This is very useful for developers when they debug. The best is to take an example. Did you try the who module at the end of Section 6.4? If so take it and check it works. Now change the line `string command = QUERY(path2who)+" "+QUERY(options2who);` to `string command = 0;`. This will create an error because we put an int into a string. If we want to do that, we have to cast it (for example, use (string) 0). If you have not done it yet, press the `More options` button in the CIF. and reload the module. Check that the `Global Variables` –> `show_internals` option is set to yes, and try your module. You will have an error which should look like this:

```
Caudium version: Caudium (Caudium/1.2.0)
Requested URL: /who

Error: Sprintf: Wrong type for argument 2: expected string, got int.
../local/modules/who.pike:76:
CaudiumModule(Who,My first virtual server)->find_file("",object)
base_server/configuration.pike (version 1.91):1587:
Configuration(My first virtual server)->low_get_file(object,0)
base_server/configuration.pike (version 1.91):1779:
Configuration(My first virtual server)->get_file(object,0)
base_server/configuration.pike (version 1.91):1760:
Configuration(My first virtual server)->handle_request(object)
protocols/http.pike (version 1.71):1549: unknown function()
protocols/http.pike (version 1.71):1610:
unknown function(0,"GET /who HTTP/1.1\r\nHost: localhost\r\nUser-Agent:
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.8)
Gecko/20020214\r\nAccept: text/xml,application/xml,
application/xhtml+xml,text/html;q=0.9,text/plain"+[246])
/usr/local/pike/7.2.262/lib/modules/Stdio.pmod/module.pmod
(version 1.114):683:
Stdio.File("socket", "127.0.0.1 1260", 777 /* fd=-1 */)
->__stdio_read_callback()
```

This seems awful but it is not. The first line is the error in itself:

```
"Error: Sprintf: Wrong type for argument 2: expected string, got int."
The next line "../local/modules/who.pike:76:
CaudiumModule(Who,My first virtual server)->find_file("",object)"
is the program (../local/modules/who.pike at line 76)
```

where the error occurred. `find_file` is the name of the function where the error occurred and you have also the arguments given to it. If you use the source, you see mixed find_file(string path, object id). So here path="" and id=object [3]. Next line is the function (`low_get_file` in `configuration.pike`) that has called `find_file` in `who.pike`. You also have its arguments and so on. This backtrace is very useful when the error doesn't come directly from your code but from another code before.

6.6. How to print something to debug log file

With Caudium you can output something to the web page or to the debug log file located in `../logs/debug/default.*`. This way the end−user will not see any line you can output in your debug log file. Sending output to the debug log file is simple, just write to stdout:

```
write("my message to log file\n");
```

It is also usually useful to use `sprintf` to format what you want to output:

```
int i = 2; write(sprintf("i=%d\n", i));
```

This line will output `i=2`, but it is better when you output array or mapping, as Pike is able to print them in a human−comprehensible format:

```
array a = ({ "test", "test2", 2 }); write(sprintf("a=%O\n", a));
```

Which will output:

```
a = ({ /* 2 elements */
    "test",
    "test2"
})
```

the %O format is very useful since it can output any type from int to mapping. The only type you can't format is object.

Chapter 7. How to help the Caudium community

Caudium is a great product, but even with the best programmers in the world it would be nothing if you didn't help us. This doesn't necessarily mean working for Caudium twelve hours a day but thinking of it for a few seconds.

The next paragraph is mainly for users who just want to help us without getting involved too much.

7.1. How to promote Caudium

Caudium, and Roxen problems are not technical. Instead, they are advertising and marketing. Caudium needs to become recognized. It could the best product in the world, but if nobody knows, it will not be used. To help Caudium to become recognized for its quality there are some simple things we can do together:

- Try it, and give us feedback about your experience.
- Write some documentation. Without documentation, a project is useless.
- Recommend Caudium to some people you know. You don't really have to tell them to try it now or they will die, just tell him about it. If other people tell them the same thing, chances are that they will install it. When these people install Caudium, they will look for documentation.
- Mention Caudium in newsgroups/web sites telling them what you think of it.
- Translate documentation.

If you do this, you will help promote Caudium's success. Remember, one of Caudium disadvantages is that the community is too small right now.

The next paragraph is for people who want to get involved a little more, but don't have lot of time, and/or are not technically minded.

7.2. How to write documentation

There are two types of documentation needed for Caudium:

- Press articles.

  You can write press articles about Caudium.

  Note for French writers:

  You can contact us for that.

  This type of documentation is usually not really helpful since few in the community will read it. It is useful mainly for advertising (even if you made a very technical article, this is useful).
- Documentation available from Internet.

  This type of documentation is a must; Caudium can't exist without it. This documentation will help Caudium's newbies, and will also be useful for developers. One good way to start with Caudium is using the Roxen 1.3 documentation. The developers should also see the autodoc. These are in−line documents made in the sources, and available as formatted HTML output at http://caudium.net/.

There is also a lack for a more in−depth documentation for developer. Examples include how to code non−blocking sockets, how to use a backtrace, how to use the `do_output_tag`, and when should I put *`thread_safe=1`*. I remember spending hours trying to understand some of these things on IRC without any documentation available. This sort of documentation should be written by one of the top developers.

This documentation is written using XSLT, but if you don't know it, that's not a problem. Give us whatever format you want, we will translate it. It is the same for the language. If you don't speak English, you can write it in your own language and we will translate. We will also proof−read your work carefully in order to correct any typos or inaccuracies you might leave.

To write this type of documentation, contact the person in charge of the documentation project, ice at caudium dot net.

7.3. How to get a CVS account

Unlike other projects, it's relatively not much hassle to get access to Caudium cvs tree. Even if you are a beginner you can have cvs access; we just ask you not to change Caudium's core at the beginning.

To get a CVS account, come on IRC or send a mail to kiwi at caudium dot net and explain your plan.

7.4. How to test Caudium

There is a test suite for Caudium written in Expect which uses TCL and DejaGNU. If you want to use it, fetch it from Caudium's CVS repository with the `−P` testsuite option (the information on this CVS repository is on http://caudium.net/.) Note that this test suite is not maintained, and needs some work.

7.5. How to send a bug report

To send a bug report, copy/paste the error from Caudium's log file (including backtrace, see Section 6.5), your configuration, and the context in which it happened.

Go to http://caudium.net/ and choose bug tracking on the left and follow the instructions.

Chapter 8. Revision History/Credits/The End

8.1. Revision History

- Version 2.2 : September 20, 2002

  Added images and icons.
  Better use of Docbook.
  Updates in the FreeBSD ports section.
  Few more fixes.
- Version 2.1 : June 26, 2002

  Little language and cosmetic corrections.
  Added the script for image selection in the your first Pike script.
  Corrections in the FreeBSD ports section.
  The file is now well−formed XML.
- Version 2.0 : June 14, 2002

  LDP Review of documentation.
  Added cross reference to full GFDL in Appendix A.
  Converted to Docbook XML 4.1.2.
  Removed editing notations that are corrected.
  Corrected links to LDP (Linux Documentation Project).
  Made sentence level corrections−spelling, grammar, etc.
- Version 1.0 : June 6, 2002

  Initial Release
  Submitted to the LDP
- Version 0.91 : May 22, 2002

  The SGML source is now the authoritative source
  The SGML source is now correctly indented
  Many little SGML changes in the document
  Added the Upgrading Caudium paragraph
  Added a note about the start script config−dir option
- Version 0.9 : May 7, 2002

  Better HTML conformance to avoid Caudium crash on caudium.info.
  Inserting the notes from Kiwi, Grendel and other into the document.
  Removed all TODO and #define sections.
  Completed the test suite section.
  Added the contributors section.
  Now docbook2ps and docbook2pdf correctly parse the SGML file.

- Version 0.05 : April 4, 2002

  This HOWTO is in Docbook format. Thanks to Thomas Marteau.
- Version 0.04 : March 26, 2002

  Added Introduction and license.
  Added a note in "Your first pike script".
  Corrected typo in paragraph number.
  Re−organization of the document.
- Version 0.03 : March 22, 2002

  More Caudium disadvantages and some useful corrections from people at the Caudium general mailing list.

  Answers:

  How to get UltraLog working.
  How to benchmark a web server.
  How to use a backtrace.
  How to print something to debug log file.
  How to get CAMAS from cvs/source.
- Version 0.02 : March 18, 2002

  Minor correction in other answers.

  Answers:

  How to run Caudium as a non−privileged user.
  How to secure Caudium.
  How to tune Caudium for best performance.
  How to get packages for Debian GNU/Linux.
  How to get packages for FreeBSD.
  How to get packages for Solaris.
  How to use your own fonts.
  How to help Caudium Community.
- Version 0.01 : March 11, 2002

  Initial revision. Plain text format.

8.2. Credits and contributors

Here is the list of people that helped me in one way or another to get this HOWTO written:

Joe Follansbee <joef at compelinteractive dot com>
Grammar, spelling and syntax cleanup.
Bill Welliver <Bill.Welliver at fairchildsemi dot com>

Minor changes.
Chris Davies <mcd at daviesinc dot com>
Major contribution to the Caudium's disadvantages, many little fixes and spelling corrections.
Martin Friese <mf at bauko dot bv dot tu−berlin dot de>
Update in the platform section.
Xavier Beaudouin <kiwi at oav dot net>
All tuning sections, cosmetics changes.
Marek Habersack <grendel at caudium dot net>
Minor fixes and latex conversion.
Thomas Marteau <marteaut at tuxfamily dot org>
Minor fixes and SGML conversion.
John Wenger <JohnWenger at EarthLink dot Net>
Editing, review and comments.

8.3. The End

Thanks for reading. I hope this HOWTO will help you discover Caudium's power. And remember this quotation:

> Documentation is like sex: when it is good, it is very, very good; and when it is bad, it is better than nothing.
>
> −−Dick Brandon

If you have any questions about this documentation, contact me at `<vida at caudium dot net>` or the Caudium general mailing list at `<general at oav dot net>`

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written 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.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

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. (For example, 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.

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 "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human modification. Opaque formats include PostScript, PDF, 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 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.3. 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.

A.4. COPYING IN QUANTITY

If you publish printed copies 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 publicly−accessible computer−network location containing a complete Transparent copy of the Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. 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.

A.5. 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 less than five).
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", and 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. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, 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 as "Endorsements" or to conflict in title with any Invariant Section.

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.

A.6. 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.

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."

A.7. 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.

A.8. 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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self–contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.9. 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.10. 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.

A.11. 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.

A.12. 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 (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front−Cover Texts being LIST, and with the Back−Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front−Cover Texts, write "no Front−Cover Texts" instead of "Front−Cover Texts being LIST"; likewise for Back−Cover Texts.

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.

Notes

[1] To identify you can use lsof(8). If you want to know which programs listen on port 80 just issue the following command as root

> You need to be root if lsof has been compiled with the HASSECURITY option which is the default for some GNU/Linux distributions

:

```
# lsof −i TCP:80
```

Here is the result:

```
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
caudium 1001 root   12u  IPv4   3993       TCP *:www (LISTEN)
```

[2] In fact you can... just compile PHP for Caudium :)

[3] Pike can't display contents of an object but can display any other types.