Writing Perl Modules for CPAN

SAM TREGAR

Apress™

ISBN (pbk): 1-59059-018-X

Printed and bound in the United States of America 12345678910

Technical Reviewers: Jesse Erlbaum and Neil Watkiss

Editorial Directors: Dan Appleman, Gary Cornell, Jason Gilmore, Simon Hayes, Karen Watterson, John Zukowski

Managing Editor and Production Editor: Grace Wong

Project Managers: Erin Mulligan, Alexa Stuart

Copy Editor: Ami Knox

Proofreader: Brendan Sanchez

Compositor: Susan Glinert

Indexer: Valerie Perry

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

Marketing Manager: Stephanie Rodriguez

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States, phone 1-800-SPRINGER, email orders@springer-ny.com, or visit http://www.springer-ny.com.

Outside the United States, fax +49 6221 345229, email orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax: 510-549-5939, email info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section. You will need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 1

CPAN

The Comprehensive Perl Archive Network (CPAN) is an Internet resource containing a wide variety of Perl materials—modules, scripts, documentation, and Perl itself—that are open source and free for all to download and use. CPAN has more Perl modules available than any other resource, including modules for almost every conceivable task from database access to GUI development and everything in between. The primary gateway to CPAN is `http://www.cpan.org`.

No other programming community has a resource like CPAN. CPAN enables the Perl community to pool its strength and develop powerful solutions to difficult problems. Once a module is available on CPAN, everyone can use it and improve it. Only the most unusual Perl project needs to start from scratch.

CPAN is more than just a repository—it's a community. The modules on CPAN are released under open-source licenses, and many are under active development. Modules on CPAN often have mailing lists dedicated to their development with hundreds of subscribers.

As the name implies, CPAN is a network. CPAN servers around the world provide access to the collection. See the "Network Topology" section later in this chapter for details.

Why Contribute to CPAN?

CPAN thrives on the time and energy of volunteer programmers. You may be surprised that so many talented programmers are willing to work for free. Some CPAN programmers aren't actually donating their time—they're being paid to work on CPAN modules! This is certainly the minority, so let's look at some other reasons to join the CPAN community.

The Programmer's Incentive

For the lone programmer, contributing to CPAN is an excellent way to show the world your programming savvy. A programmer's resume is only an introduction; a smart employer wants proof. This can be hard to provide if all your work has been on closed-source projects. Open-source software is easy to evaluate—if you're good, employers will know it immediately. There's nothing quite like walking into

an interview and having the programmer across the table suddenly realize he's been using your code for the past two months.

As software reaches higher levels of maturity and complexity, it is less and less realistic for a programmers to "go it alone." Today, conscientious and talented programmers first look to CPAN to provide a shortcut in their development process—and the *best* programmers contribute their work to CPAN, so that others may benefit. Tomorrow, it may even be considered a lack of professionalism to *not* start your software development efforts with a search through the CPAN repository.

By writing code for CPAN, you'll come into contact with other highly talented Perl programmers. This has been a great help to me personally—the many bug reports and suggestions I've received over the years have helped me improve my skills. With Perl, there's *always* more than one way to do it, and the more of them you master, the better.

The Business Incentive

Just as contributing to CPAN enhances a programmer's resume, so can a business benefit by association with popular Perl modules. Contributing your modules to CPAN can have the effect of establishing a standard around your practices. This makes answering the perennial question "Why aren't you using *[Java, C++, ASP, PHP]*?" much easier.

Some of the world's best programmers are open-source programmers. By actively supporting CPAN, you improve your hiring ability in the competitive market for Perl experts.

The Idealist's Incentive

For the idealist, contributing to CPAN is a good way to help save the world. CPAN is open to everyone—multinational corporations and tiny nonprofits eat at the same table. When you donate your work to CPAN, you ensure that your work will be available to anyone who needs it. Furthermore, by putting your work under a free software[1] license you can help convince others to do the same; when they make changes to your code, they'll have to release them as free software.[2]

1. See `http://www.fsf.org` for more information about free software.
2. With some notable exceptions—see the "Choosing a License" section of Chapter 4 for more details.

CPAN History

The idea for CPAN, a single comprehensive archive of all things Perl, was first introduced in 1993 by Jared Rhine on the perl-packrats mailing list.[3] The concept derived from the Comprehensive TeX Archive Network (CTAN). At this point a number of large Perl archives were maintained on various FTP sites around the world. It was widely agreed that there would be many advantages to collecting all the available Perl materials in one hierarchy; however, the discussion died without producing a working version.

In early 1995 Jarkko Hietaniemi resurrected the idea and began the monumental task of gathering and organizing the entire output of the Perl community into a single tree. Six months later he produced a working "private showing." This CPAN was essentially a sorted, classified version of the contents of every Perl archive on the Internet.

However, a critical piece was missing—a way for Perl authors to upload their work and have it automatically included in CPAN. Andreas Köenig came to the rescue by creating the **P**erl **A**uthor **U**pload **SE**rver (PAUSE). PAUSE automatically builds the `authors` and `modules-by` directories that form the bulk of content on CPAN (86.5 percent at present).

With PAUSE in place, CPAN was nearly complete. After two months of testing and fixing with the help the perl-packrats, Jarkko released CPAN to the world as the Self-Appointed Master Librarian. The master server was set up at FUNet, where Jarkko worked as a systems administrator, which is where it remains today. From then on CPAN played a central role in the growth of the Perl community.

Network Topology

CPAN is composed of servers spread across the globe (over 200 as I write). Every server provides access to the same data. Figure 1-1 shows a map of CPAN servers. You can explore the CPAN network interactively at `http://mirror.cpan.org`.

3. he perl-packrats list, active from 1993 to 1996, was formed to discuss archiving Perl. Mailing list archives can be found at `http://history.perl.org/packratsarch/`.

Figure 1-1. World map from `http://mirrors.cpan.org` *showing CPAN server locations*

CPAN is modeled on a hub-and-spokes topology, shown in Figure 1-2. At the center of the CPAN network is the main CPAN server, ftp.funet.fi, in Finland. Most of the CPAN servers mirror this main server directly. To *mirror* is to maintain a synchronized copy of the files between two machines. CPAN servers use either FTP or rsync to automatically mirror files.

Modules enter CPAN through a system called PAUSE, short for the **P**erl **A**uthor **U**pload **SE**rver. I'll provide more details about PAUSE in Chapter 4.

Since CPAN is a network, you can choose a mirror close to you that may offer faster download times than `http://www.cpan.org`. At `http://mirror.cpan.org` you'll find a search facility that enables you to search for mirrors by country.[4]

4. Of course, the fastest way to access CPAN is by running your own mirror. See `http://www.cpan.org/misc/cpan-faq.html#How_mirror_CPAN` for details.

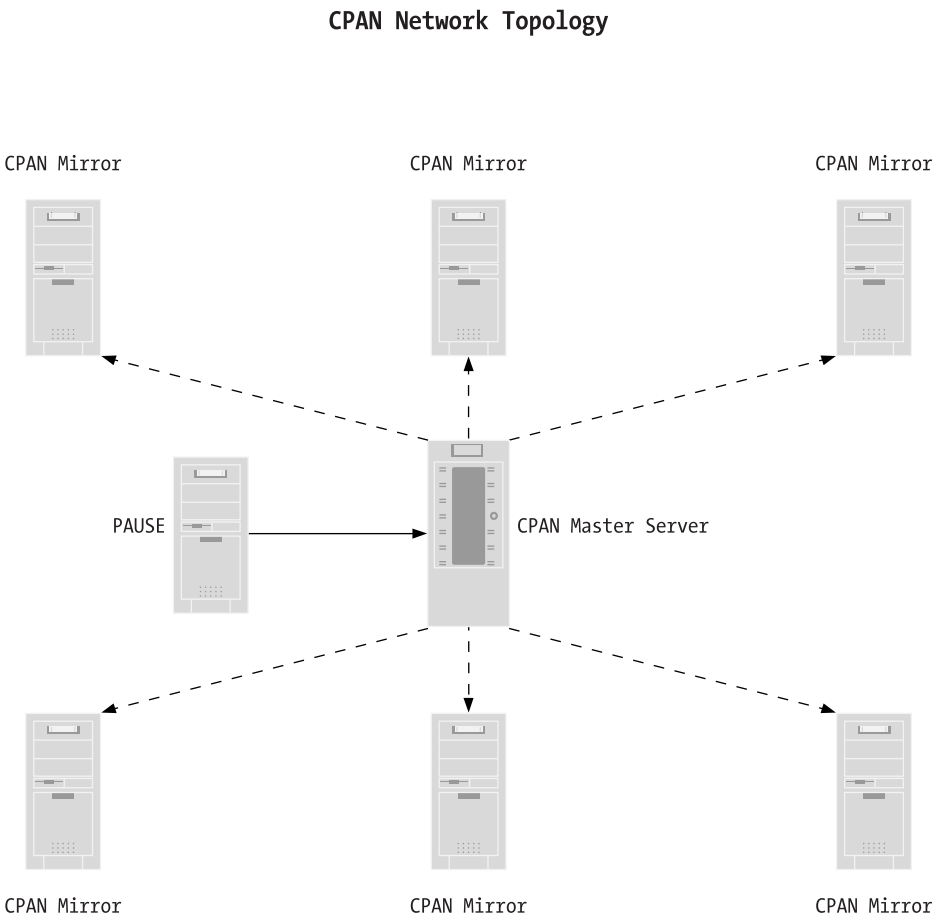

Figure 1-2. The CPAN Network Topology

Browsing CPAN

If this is your first time visiting CPAN, the first thing you should do is have a look around. On the entry screen (Figure 1-3) you'll find links to each section of the CPAN collection—modules, scripts, binaries, the Perl source, and other items. Also available are links to documentation about CPAN; if you still have questions after finishing this chapter, then you should give them a look.

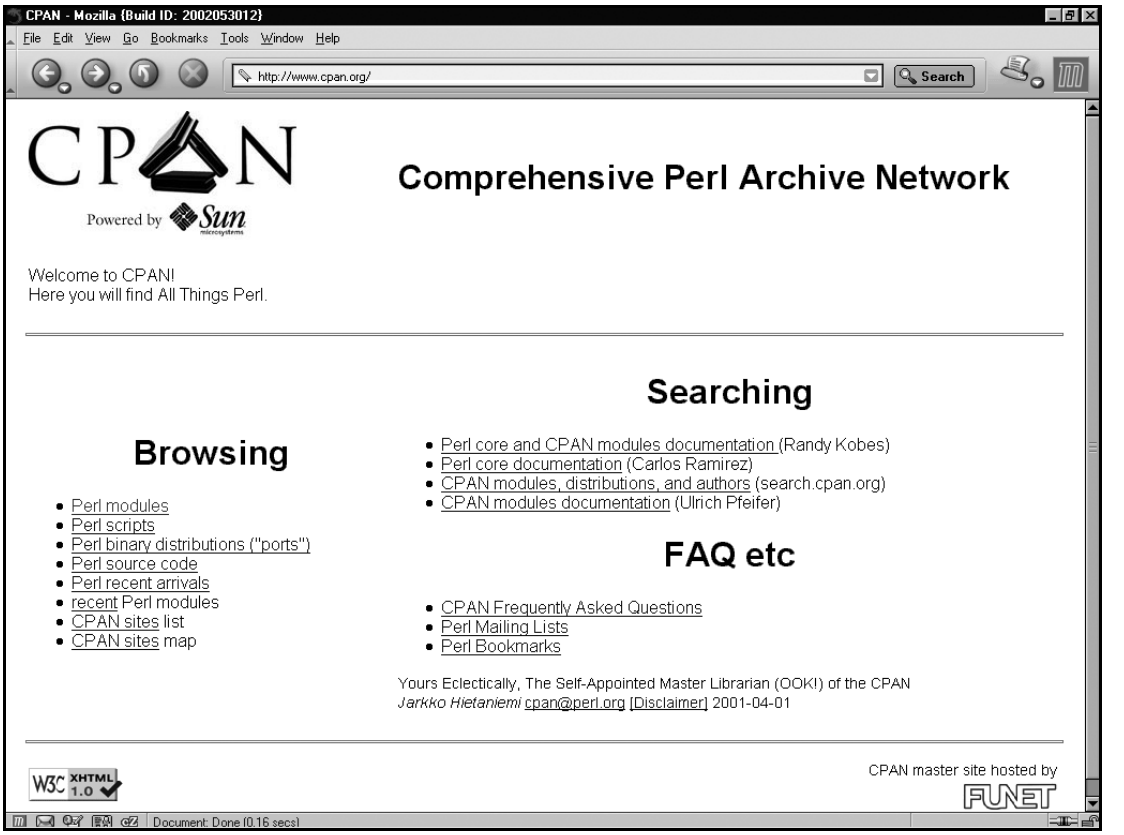

Figure 1-3. Entry screen for `http://www.cpan.org`

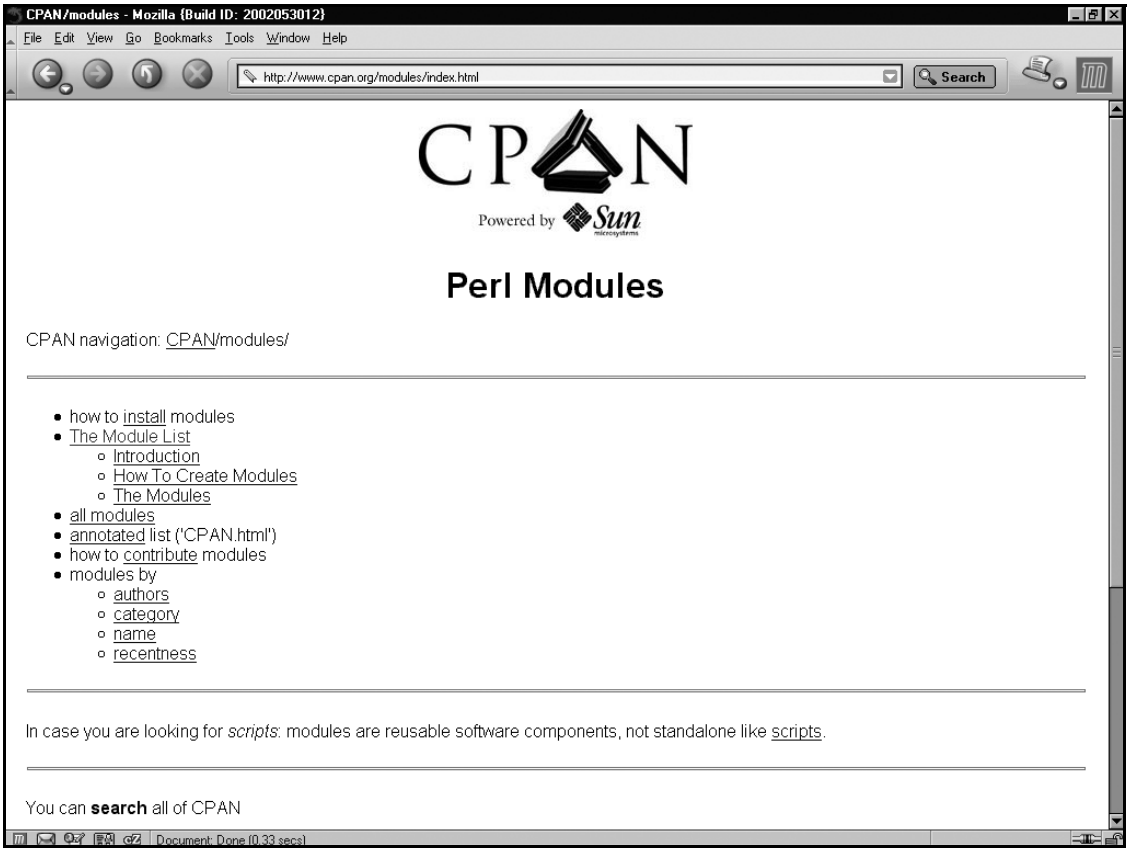

Figure 1-4. CPAN modules menu

I suggest you begin by entering the modules section of CPAN. This is by far the most useful area of the site and also the subject of this book. It's good to know where to find Perl, but you probably already know a thing or two about that if you're thinking about writing CPAN modules. Figure 1-4 shows the CPAN modules menu, where you'll find a number of different ways to navigate through the module collection.

The Module List

The Module List is a semi-manually maintained list of most of the Perl modules on CPAN. A section of the Module List is shown in Figure 1-5.

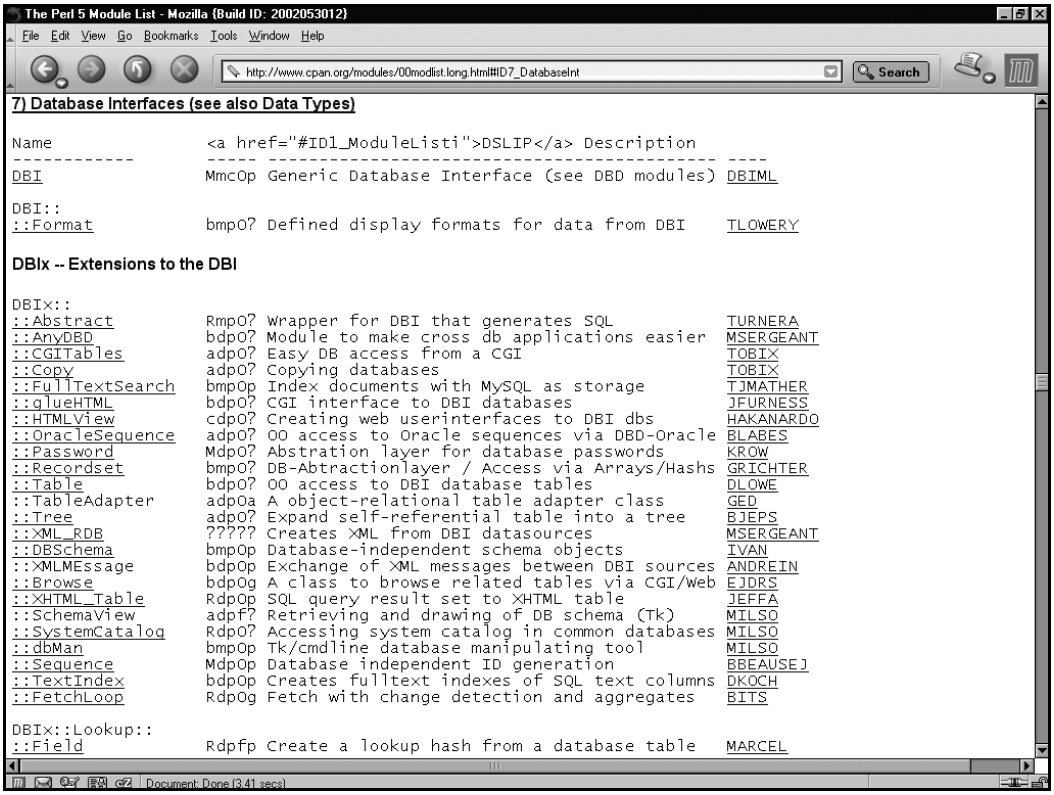

Figure 1-5. The start of Database Interfaces section in the Module List

In many ways, its function has been superseded by the newer search interfaces detailed later in this chapter, but it does have some unique features that can be helpful. First, it organizes the modules into categories by function. These categories are listed here:

Module List Categories

Perl Core Modules, Perl Language Extensions, and Documentation Tools

Development Support

Operating System Interfaces, Hardware Drivers

Networking, Device Control, and Interprocess Communication

Data Types and Data Type Utilities

Database Interfaces

User Interfaces

Interfaces to or Emulations of Other Programming Languages

File Names, File Systems, and File Locking

String Processing, Language Text Processing, Parsing, and Searching

Option, Argument, Parameter, and Configuration File Processing

Internationalization and Locale

Authentication, Security, and Encryption

World Wide Web, HTML, HTTP, CGI, MIME, and so on

Server and Daemon Utilities

Archiving, Compression, and Conversion

Images, Pixmap, and Bitmap Manipulation

Mail and Usenet News

Control Flow Utilities

File Handle, Directory Handle, and Input/Output Stream Utilities

Microsoft Windows Modules

Miscellaneous Modules

Interface Modules to Commercial Software

Bundles

Secondly, each listing contains a DSLIP code that can give you some information about the status of the module. *DSLIP* stands for Development Stage, Support Level, Language Used, Interface Style, and Public License. For example, a DSLIP code of *bmpOp* specifies that the module is in beta testing (*b*), is supported by a mailing-list (*m*), is written in pure Perl (*p*), has an object-oriented interface (*O*) and is licensed under the same license as Perl (*p*). Table 1-1 lists the various DSLIP codes.

Table 1-1. Module List DSLIP codes

D-Development Stage	
I	Idea
c	Under construction
a	Alpha testing
b	Beta testing
R	Released
M	Mature
S	Standard, comes with Perl 5
S-Support Levels	
m	Mailing list
d	Developer
u	Usenet newsgroup comp.lang.perl.modules
n	None
L-Language Used	
p	Perl-only
c	C and Perl
h	Hybrid, written in Perl with optional C code
+	C++ and Perl
o	Perl and another language other than C or C++
I-Interface Style	
f	Plain functions
O	Object oriented
h	Hybrid, object, and function interfaces available
r	Unblessed references or ties
n	None

Table 1-1. Module List DSLIP codes (Continued)

P-Public License	
p	Standard Perl license (Artistic and GPL hybrid)
g	GPL (GNU General Public License)
l	LGPL (GNU Lesser General Public License)
b	The BSD License
a	Artistic license
o	Other (but distribution allowed without restrictions)

The biggest problem with the Module List is that it is incomplete, although this situation may be improved in the future.

Alternative Browsing Methods

An alternative to browsing the Module List is the "modules by" listings. You can browse modules grouped by author, by category, by name, and by recentness. The advantage to this method is that it deals directly with the directory structure of CPAN and as a result all available modules are accessible.

By Author

Upon entering the Modules By Author view, you see a directory listing with what appears to be a directory for every author on CPAN. This is misleading—the list you're seeing is a relic of the past. When CPAN started every author received an entry in this directory, but there's a limit to how many subdirectories a single directory can efficiently contain. These days there are far too many authors on CPAN to house them all in one directory, so CPAN switched to a multilevel hierarchy for storing author directories, which is used today.

To see the real list, open the file `00whois.html`. There you'll find three pieces of information for each author—his or her CPAN ID, his or her full name, and his or her e-mail address. A CPAN ID is a unique identifier for CPAN authors—I'll show you how to apply for one in Chapter 5. If you click an author's CPAN ID,[5] you'll be taken to that author's CPAN directory, which contains all the modules he or she has uploaded to CPAN. Some authors have registered Web sites for themselves, and you can click their full names to visit these.

5. Some CPAN authors do not have CPAN directories. Their IDs will not be links.

By Category

The By Category view brings you to a directory hierarchy based on the categories in the Module List, listed earlier in this chapter. Inside each category you have an interface similar to the Module By Name interface described next.

By Name

Navigating CPAN modules by name allows you to traverse the module names directly, where each :: is translated into a path separator. This can be helpful when you know part of the name for the module you're looking for and need to see a list of possibilities. If you know the exact name of a module, then the search interface described later in this chapter is a faster alternative.

By Recentness

The By Recentness view shows you the most recent 150 uploads to CPAN. The format is a bit nicer than the Recent Arrivals list available on the opening screen, but it's not as nice as the format provided by `http://search.cpan.org`.

Searching CPAN

CPAN also sports a variety of search engines. Currently, the most useful is `http://search.cpan.org` (see Figure 1-6 for the entry screen). Not only does this search engine provide search capabilities, it also serves HTML versions of module documentation and gives access to a pleasantly formatted list of recently updated modules. This enables you to evaluate a group of modules without the trouble of installing them.

To use the search engine, just type a word in the search box and click the Search button. You can also enter a regular expression or choose a specific part of CPAN if you need to narrow your search. When you find a module that sounds interesting, just click the name, and you'll be brought to a details screen where you can view the module documentation.

The search interface also includes interfaces that mimic features offered by `http://www.cpan.org`. You can browse by category and see a list of recently uploaded files with an arguably prettier interface. You should try both interfaces and choose the one you like the best.

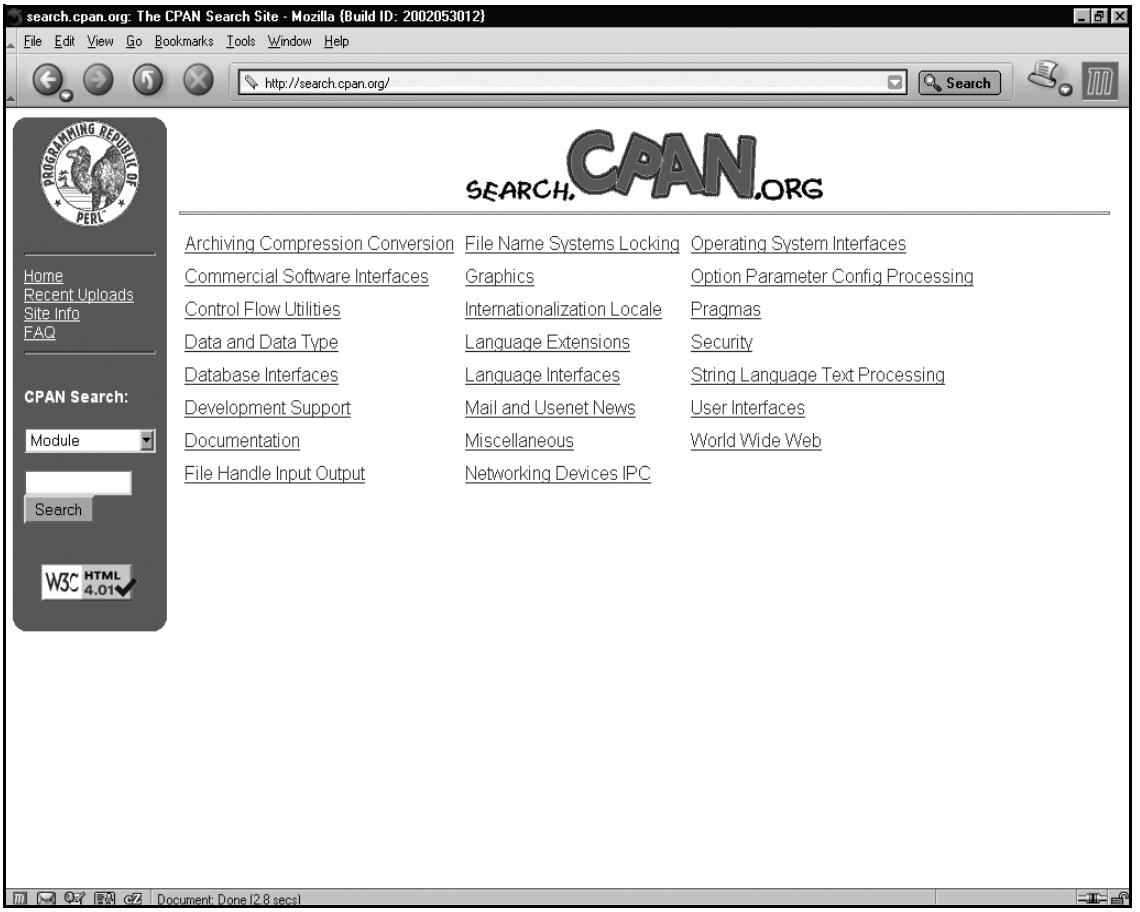

Figure 1-6. `http://search.cpan.org` *entry screen*

Installing CPAN Modules

So, you've found the module you've been searching for. Now you'll need to install it. And, like many things in Perl, TMTOWTDI.[6] The sections that follow discuss the two main installation methods: the easy way and the hard way.

The Easy Way

I'll start with the easy way—if you encounter problems, you should consult the "The Hard Way" section later in this chapter.

6. There's more than one way to do it.

Recent versions of Perl come with a module called CPAN,[7] which as you might have guessed is used to access the contents of CPAN. The CPAN module makes installing CPAN modules incredibly easy. It downloads modules from CPAN and automatically follows their dependencies, saving you a lot of work (which you'll learn all about in the upcoming section, "The Hard Way").

To get started with the CPAN module, enter the following command:

```
# perl -MCPAN -e shell
```

If you're using a UNIX system and want to install modules system-wide, you'll have to run this command as the *root* user. It is possible to use the CPAN module as a normal user, but you won't be able to install modules into the system.

The first time you run this command the CPAN module will ask you a series of questions:

```
# perl -MCPAN -e shell

CPAN is the world-wide archive of perl resources. It consists of about
100 sites that all replicate the same contents all around the globe.
Many countries have at least one CPAN site already. The resources
found on CPAN are easily accessible with the CPAN.pm module. If you
want to use CPAN.pm, you have to configure it properly.

If you do not want to enter a dialog now, you can answer 'no' to this
question and I'll try to autoconfigure. (Note: you can revisit this
dialog anytime later by typing 'o conf init' at the cpan prompt.)

Are you ready for manual configuration? [yes]
```

Each question has a default answer in square brackets. In most cases the default will be correct and you can just press Enter to continue. One important question to look for is this one, about following prerequisites:

```
The CPAN module can detect when a module that which you are trying to
build depends on prerequisites. If this happens, it can build the
prerequisites for you automatically ('follow'), ask you for
confirmation ('ask'), or just ignore them ('ignore'). Please set your
policy to one of the three values.

Policy on building prerequisites (follow, ask or ignore)? [ask]
```

7. Written and maintained by Andreas Köenig

The default, `ask`, is the most conservative setting, but you should consider answering `follow` since this will greatly ease the task of installing modules with lots of dependencies.

The CPAN modules uses various external programs, and you'll be asked to confirm their location:

```
Where is your gzip program? [/bin/gzip]
```

If you don't want the CPAN module to use a particular external program type a space and press Enter. This can be useful if you know a program is broken on your system or won't be able to perform its task.

Towards the end of the questions, the CPAN module will present you with a choice of which mirrors to use. First, you'll identify your continent:

```
(1) Africa
(2) Asia
(3) Central America
(4) Europe
(5) North America
(6) Oceania
(7) South America
Select your continent (or several nearby continents) []
```

then country:

```
(1) Canada
(2) Mexico
(3) United States
Select your country (or several nearby countries) []
```

and finally you'll select several mirrors from a list:

```
(1) ftp://archive.progeny.com/CPAN/
(2) ftp://carroll.cac.psu.edu/pub/CPAN/
(3) ftp://cpan.cse.msu.edu/
...
Select as many URLs as you like,
put them on one line, separated by blanks []
```

Make sure you pick more than one since many mirrors have limits on the number of people that can use them at one time. Also, not all mirrors are equally up-to-date. To make the best possible picks, you should visit `http://mirror.cpan.org`, where you can view a profile of each mirror including how up-to-date they are.

The very first thing you should do after configuring the CPAN module is install the newest version of the CPAN module and reload it. You can do that with these commands:

```
cpan> install CPAN
cpan> reload CPAN
```

This will save you the trouble of bumping into bugs in the CPAN module that have been fixed since the version that comes with Perl came out. In particular, older versions of the CPAN module had a nasty habit of trying to upgrade Perl without asking permission. The examples in this book are based on version `1.59_54` of the CPAN module, but using the newest version is always a good idea.

TIP If you're having trouble connecting to CPAN using the CPAN module, you might need to manually install the Net::FTP module. See the section that follows on installing modules the hard way for details on how to do this.

After that, your next stop should be the CPAN bundle. The CPAN bundle contains a number of modules that make the CPAN module much easier to use and more robust. To install the bundle, use this command:

```
cpan> install Bundle::CPAN
```

NOTE See the "Bundles" section later in this chapter to find out how Bundles work.

Now you're ready to install modules. For example, to install the CGI::Application module,[8] you would enter the following command:

```
cpan> install CGI::Application
```

And the CPAN module will handle downloading the module, running module tests, and installing it. If CGI::Application requires other modules, then the CPAN module will download and install those too.

8. Described in Chapter 11

The CPAN module is versatile tool with myriad options and capabilities. While in the CPAN shell, you can get a description of the available commands using the `help` command. Also, to learn more about the module itself, you can access the CPAN documentation, using the `perldoc` utility:

```
$ perldoc CPAN
```

The Hard Way

The CPAN module may not be right for you. You may be behind a firewall or you might prefer more control over the module installation process. Also, some CPAN modules, usually older ones, aren't written to work with the CPAN module. If this is the case, then you'll need to install modules the hard way. Put on your opaque sunglasses and grab your towel.

Location

First, find the module you want to download on the CPAN server near you. An easy way to do this is by using the CPAN Search facilities described earlier. The file you're looking for will end in either `.tar.gz` or `.zip`. CPAN modules have version numbers, and there will usually be a list of versions to choose from. You'll generally want to choose the highest version number available. Download the module file and put it in a working directory on your local machine.

Decompression

These files are compressed, so the first thing you'll need to do is uncompress them to get at their contents. Under UNIX systems this is usually done with the `tar` and `gzip` utilities:

```
$ gzip -dc ModuleNameHere.tar.gz | tar xvf -
```

Under Windows you can use tools such as WinZip, available at `http://www.winzip.com`, or install a Windows port of the GNU utilities such as CygWin, which includes `tar` and `gzip`. CygWin is available at `http://cygwin.com`.

Build

Now that you've unpacked the module, you need to build it. Enter the directory created by unpacking the compressed module file. It's usually named the same as the compressed file but with the `.tar.gz` or `.zip` ending removed.

If the module has no installation instructions, look for a file called `Makefile.PL`. If it exists, enter the following commands:

```
$ perl Makefile.PL
$ make
```

These commands will fail if you're missing a *prerequisite module*. A prerequisite module is a module that is needed by the module you're installing. If the module has unsatisfied prerequisites, you'll need to find the required module or modules and install them before returning to installing this module.

These commands may also fail if you're using a Microsoft Windows system, because few Windows systems have the `make` utility installed. You may need to install the CygWin toolkit I mentioned in the "Decompression" section, which offers the GNU `make` utility as an optional component. Alternately, you may have a program called `nmake`[9] or `dmake`, which can function as `make`.

Regrettably, there are some modules on CPAN that don't use the standard module packaging system. Sometimes these modules will include an `INSTALL` file containing installation instructions, or installation instructions will be contained in the `README` file.

Test

Many CPAN modules come with tests to verify that the module is working properly on your system. The standard way to run module tests is with this command:

```
$ make test
```

9. You can download `nmake` from `http://download.microsoft.com/download/vc15/Patch/1.52/W95/EN-US/Nmake15.exe`.

Install

Finally, you will need to install the module to be able to use the module in your programs. To do so, enter the following command:

```
# make install
```

You will need to be *root* to perform this step on UNIX systems.

ActivePerl PPM

If you are using Perl on a Microsoft Windows system, there's a pretty good chance you are using ActiveState's[10] ActivePerl distribution. ActivePerl is also available for Linux and Solaris. If you're using ActivePerl, then you have a utility called `PPM` that can potentially make module installation even easier than using the CPAN module. Specifically, `PPM` will install binary distributions from the `PPM` repository at ActiveState (and elsewhere). This makes installing C-based modules possible on machines without C compilers. It also alleviates the need to install `make`, `nmake`, or `dmake` as previously described.

The downside is that the ActiveState `PPM` repository isn't CPAN. It contains many of the most popular CPAN modules, but many are missing. Even worse, the modules that are present are often out-of-date compared to the CPAN versions.

Using `PPM` is a lot like using the CPAN module's shell. To get started, use this command in your system's shell:

```
ppm
```

Now you'll be presented with a `PPM` prompt. The most common command is `install`, which allows you to install modules. This command will install a (probably out-of-date) version of my HTML::Template module:

```
install HTML::Template
```

To learn more about PPM, you can use the online help facility in the PPM shell with the `help` command.

10. See `http://www.activestate.com`.

Bundles

A bundle is a module that allows you to install a list of modules automatically using the CPAN module. A bundle is simply a module in the Bundle:: namespace containing a list of modules to download; it doesn't contain other modules. A bundle can also specify the versions of the modules to be downloaded, so that it can serve as a "known-good" module set.

To use a bundle, simple install it with the CPAN module. For example, to install Bundle::CPAN, enter the following:

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

There are bundles available for many popular module groups: Bundle::LWP, Bundle::DBI, and Bundle::Apache, for example. To get a list of all bundles on CPAN, use the bundle search command `b` in the CPAN shell:

```
cpan> b /Bundle::/
Bundle          Bundle::ABH         (A/AB/ABH/Bundle-ABH-1.05.tar.gz)
Bundle          Bundle::ABH::Apache (A/AB/ABH/Bundle-ABH-1.05.tar.gz)
...
```

CPAN's Future

Writing about CPAN is a risky proposition, as it is under constant development. Use this chapter as a starting point and be prepared to find things a bit different than I've described them.

Summary

This chapter has introduced you to the wonderful world of CPAN. If I've done my job, by now you're interested in joining the CPAN community. The next chapter will introduce the science of building modules in Perl.