PERL - EMBEDDED DOCUMENTATION

You can embed Pod *PlainOldText* documentation in your Perl modules and scripts. Following is the rule to use embedded documentation in your Perl Code −

> *Start your documentation with an empty line, a **=head1** command at the beginning, and end it with a **=cut** command and an empty line.*

Perl will ignore the Pod text you entered in the code. Following is a simple example of using embedded documentation inside your Perl code −

```
#!/usr/bin/perl

print "Hello, World\n";

=head1 Hello, World Example
This example demonstrate very basic syntax of Perl.
=cut

print "Hello, Universe\n";
```

When above code is executed, it produces the following result −

```
Hello, World
Hello, Universe
```

If you're going to put your Pod at the end of the file, and you're using an __END__ or __DATA__ cut mark, make sure to put an empty line there before the first Pod command as follows, otherwise without an empty line before the **=head1**, many translators wouldn't have recognized the **=head1** as starting a Pod block.

```
#!/usr/bin/perl

print "Hello, World\n";

while(<DATA>){
  print $_;
}

__END__

=head1 Hello, World Example
This example demonstrate very basic syntax of Perl.
print "Hello, Universe\n";
```

When above code is executed, it produces the following result −

```
Hello, World

=head1 Hello, World Example
This example demonstrate very basic syntax of Perl.
print "Hello, Universe\n";
```

Let's take one more example for the same code without reading DATA part −

```
#!/usr/bin/perl

print "Hello, World\n";
```

```
__END__

=head1 Hello, World Example
This example demonstrate very basic syntax of Perl.
print "Hello, Universe\n";
```

When above code is executed, it produces the following result −

```
Hello, World
```

What is POD?

Pod is a simple-to-use markup language used for writing documentation for Perl, Perl programs, and Perl modules. There are various translators available for converting Pod to various formats like plain text, HTML, man pages, and more. Pod markup consists of three basic kinds of paragraphs −

- **Ordinary Paragraph**: You can use formatting codes in ordinary paragraphs, for bold, italic, code-style , hyperlinks, and more.
- **Verbatim Paragraph**: Verbatim paragraphs are usually used for presenting a codeblock or other text which does not require any special parsing or formatting, and which shouldn't be wrapped.
- **Command Paragraph**: A command paragraph is used for special treatment of whole chunks of text, usually as headings or parts of lists. All command paragraphs start with **=**, followed by an identifier, followed by arbitrary text that the command can use however it pleases. Currently recognized commands are −

```
=pod
=head1 Heading Text
=head2 Heading Text
=head3 Heading Text
=head4 Heading Text
=over indentlevel
=item stuff
=back
=begin format
=end format
=for format text...
=encoding type
=cut
```

POD Examples

Consider the following POD −

```
=head1 SYNOPSIS
Copyright 2005 [TUTORIALSOPOINT].
=cut
```

You can use **pod2html** utility available on Linux to convert above POD into HTML, so it will produce following result −

```
Copyright 2005 [TUTORIALSOPOINT].
```

Next, consider the following example −

```
=head2 An Example List

=over 4
=item * This is a bulleted list.
```

```
=item * Here's another item.
=back
=begin html
<p>
Here's some embedded HTML.  In this block I can
include images, apply <span style="color: green">
styles</span>, or do anything else I can do with
HTML.  pod parsers that aren't outputting HTML will
completely ignore it.
</p>

=end html
```

When you convert the above POD into HTML using pod2html, it will produce following result −

```
An Example List



    • This is a bulleted list.


    • Here's another item.




Here's some embedded HTML. In this block I can include images, apply styles, or do
anything else I can do with HTML. pod parsers that aren't outputting HTML will completely
ignore it.
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js