



## Perl::Critic

Josh McAdams  
perlcritic@gmail.com

### Introduction

Jeffrey Ryan Thalhammer created `Perl::Critic` as an “extensible framework for creating and applying coding standards to Perl source code”. `Perl::Critic` is a system that analyzes Perl source code and points out violations of coding standards. By default, it uses the standards from Damian Conway’s *Perl Best Practices (PBP)*, although when I don’t agree with *PBP* on some points, it’s easy for me to modify and extend those standards to my personal preferences. `Perl::Critic` runs as a stand-alone program, as a pragma, in an automated test suite, or embedded in a program.

### Jumping Right In

My module `DBIx::MyPassword` is a simple wrapper around `DBI`. It reads connection information from a comma-separated values (CSV) file. My `connect()` method overrides `DBI`’s `connect()` and does a look-up in a CSV file to get the authentication information and other options. The other half-dozen or so methods in my module are just lookups into the CSV file. It’s not a lot of code, so how bad can it be? When I run `perlcritic` from the command line, I get the output in Listing 1, which tells me I have some problems.

#### Listing 1: Running `perlcritic` from the command line

```
$> perlcritic lib/DBIx/MyPassword.pm
'return' statement with explicit 'undef' at line 75, column 2. See page 199 of PBP.
(Severity: 5)
Expression form of 'eval' at line 80, column 12. See page 161 of PBP. (Severity: 5)
```

`Perl::Critic` finds two big problems with my code. I have a `return` statement that explicitly returns an undefined value and an `eval` statement in the expression form. Each problem with my code comes with a short description and a reference to

the page in *PBP* where Damain slams my coding style (O’Reilly had better have given Jeffrey a promo copy of *PBP* for this publicity!).

I start by looking at that `return` with an explicit `undef` that I have in my code. Damain doesn’t like that I explicitly return an undefined value, but honestly, I think that’s what I should do. Unfortunately for me, I left my copy of *PBP* at work so I have to wait until tomorrow to see if he can adequately convince me that I shouldn’t return an undefined value. Or do I? I could ask `Perl::Critic` to tell me why I should change my ways; I just need to ask for more verbose output. In Listing 2, I use `perlcritic`’s `--verbosity` switch to turn it all the way up to 9.

From the output, I see that if I return `undef`, in list context I end up with a one element list, which is not what I want at all. Oh, well, in that case I’d better change my code to have an empty `return` statement, which still does what I need. I was one list context away from causing someone problems.

I explicitly asked for a verbosity of 9, which is the highest it goes. The standard for `Perl::Critic` is around 3, which gave me the output I showed first. The lowest value is 1. Each of these levels tells me more (or less) about what is wrong with my code,

or at least shows it in a different format. If none of these levels work for me, there is a `printf` style of formatting that I can use to customize the output from `Perl::Critic`. The defaults are fine with me, but have fun tweaking to your heart’s content,