PerlAccessorsConsideredHarmful

Perl-Style Accessors Considered Harmful

(a quick break-out page from the 'Accessors' section of CodingStyle).

We don't use traditional perl-style variable accessor methods very frequently (ie.

```
sub foo {
  my ($self, $val) = @_;
  if (defined $val) {
    $self->{foo} = $val;
  } else {
    return $val;
  }
}
```

Instead, the more wordy Java/C++ style is preferred:

```
sub get_foo {
  my ($self) = @_;
  return $val;
}
sub set_foo {
  my ($self, $val) = @_;
  $self->{foo} = $val;
}
```

The perl style is considered a bad idea, because it can become a no-op, if the value being passed in is 'undef'. Here's how:

- Let's say you have a perl-style accessor `$self->foo()`, which is used to access the value `$self->{foo}`.
- `$self->{foo}` is currently eq `'bar'`.
- A caller comes along with a variable $, *and wants to set the foo value to whatever's in $*. They therefore call `$self->foo($_)`.
- *The bug:* if `$_` is undef, that means that `$self->foo(undef)` is called. In a perl-style accessor, that is considered a 'get' operation instead of a 'set', so after that call, $self->{foo} is still set to `'bar'`.

In other words, it's impossible to use a perl-style accessor to set a value to 'undef', and it's easy to accidentally perform a no-op instead of a set. This has bitten us in the past.

In the Java-style accessor, the source code itself mandates whether the operation is a set or a get; the data cannot affect which operation happens. Hence, it's safer.