Log::Log4perl

Einfach loggen...

Richard Lippmann
horshack@lisa.franken.de

Logging ist völlig trivial:

```
our $debug=1;
...
print "DB öffnen\n" if $debug;
...
```

... oder manchmal:

```
our $debug=1;
...
print "DB öffnen<br />" if $debug;
...
print "DB tot? Connect!<br />"
    if $debug;
```

... in einem Modul:

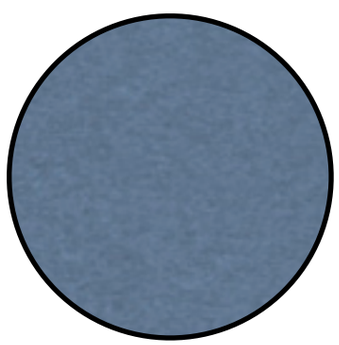
```
...
print "DB öffnen<br />" if $main::debug;
...
print "DB tot? Connect!<br />"
    if $debug;  # HUCH! Variable nicht deklariert
```

Loggen nervt

- Ich muss das Programm schreiben und habe keine Zeit mich um Logging zu kümmern
- Logausgaben verderben meinen Bildschirm
- Wichtige Dinge können nicht mehr erkannt werden
- Am Ende müssen alle print-Befehle wieder entfernt werden

Ich will doch nur...

angebot.pl
--kunde=zdf



1. Alles hat mit diesem Satz angefangen.
2. Wieviel Zeit wollen wir noch vergeuden alles noch einmal zu erfinden?
3. Haben wir diese Zeit?

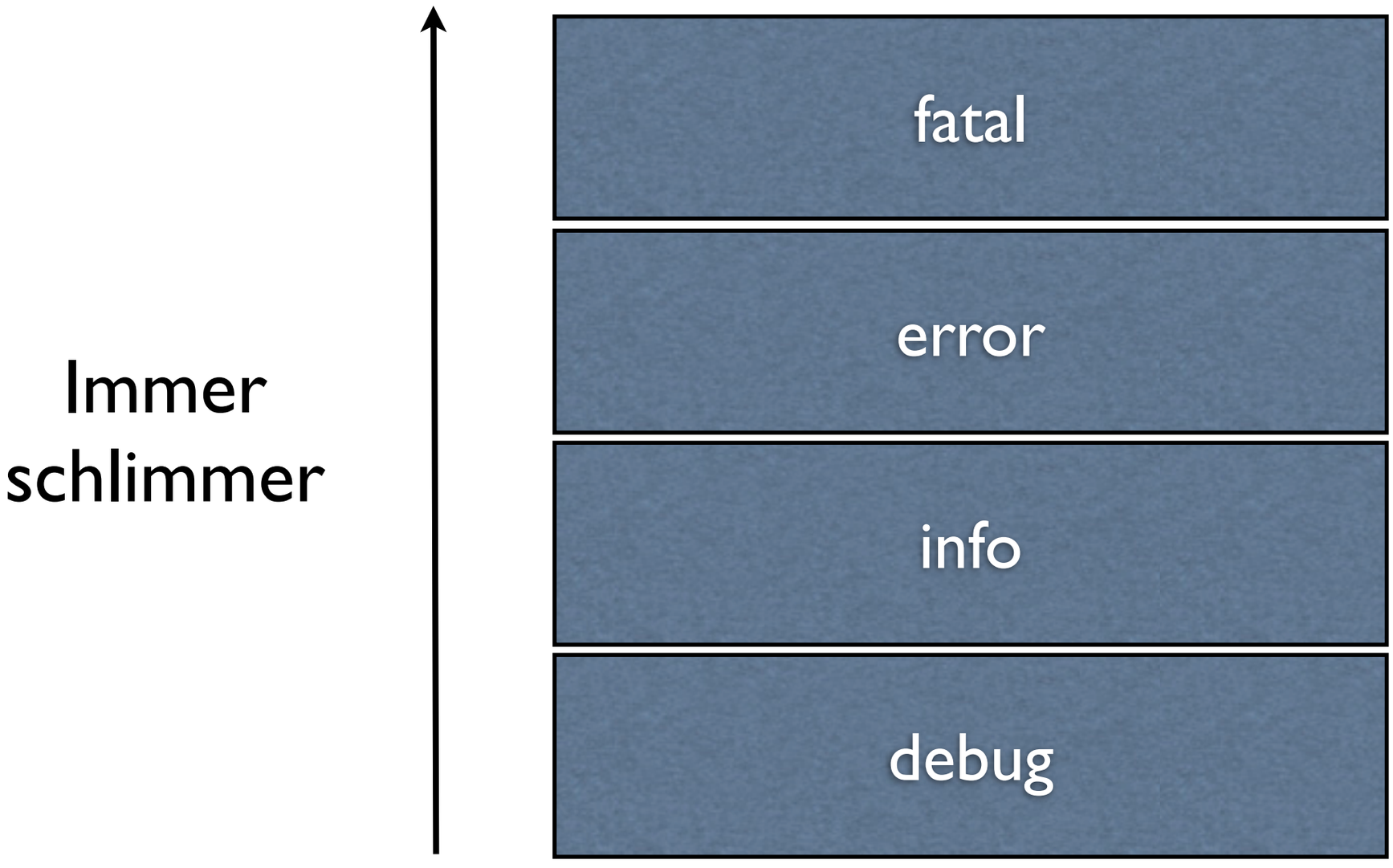
Ein kleiner Start, weil ja alles immer funktioniert brauchen wir nicht mehr.

```
# Logging starten
use Log::Log4perl qw/:easy/;
Log::Log4perl->easy_init($ERROR);
my $logger = Log::Log4perl->get_logger;

$logger->debug("$0 ist gestartet");
...
$fk = Firma::Kunden->new("...");
$alle_kunden = $fk->hole_kundenliste;
$angeb_kunde = $fk->hole_angebotsdaten(kunde => 'oberdorf');
...
$ausgabe = Firma::Ausgabe->new(
    kunde => $angeb_kunde,
    kundenliste => $alle_kunden,
    ausgabeart => ... );
```

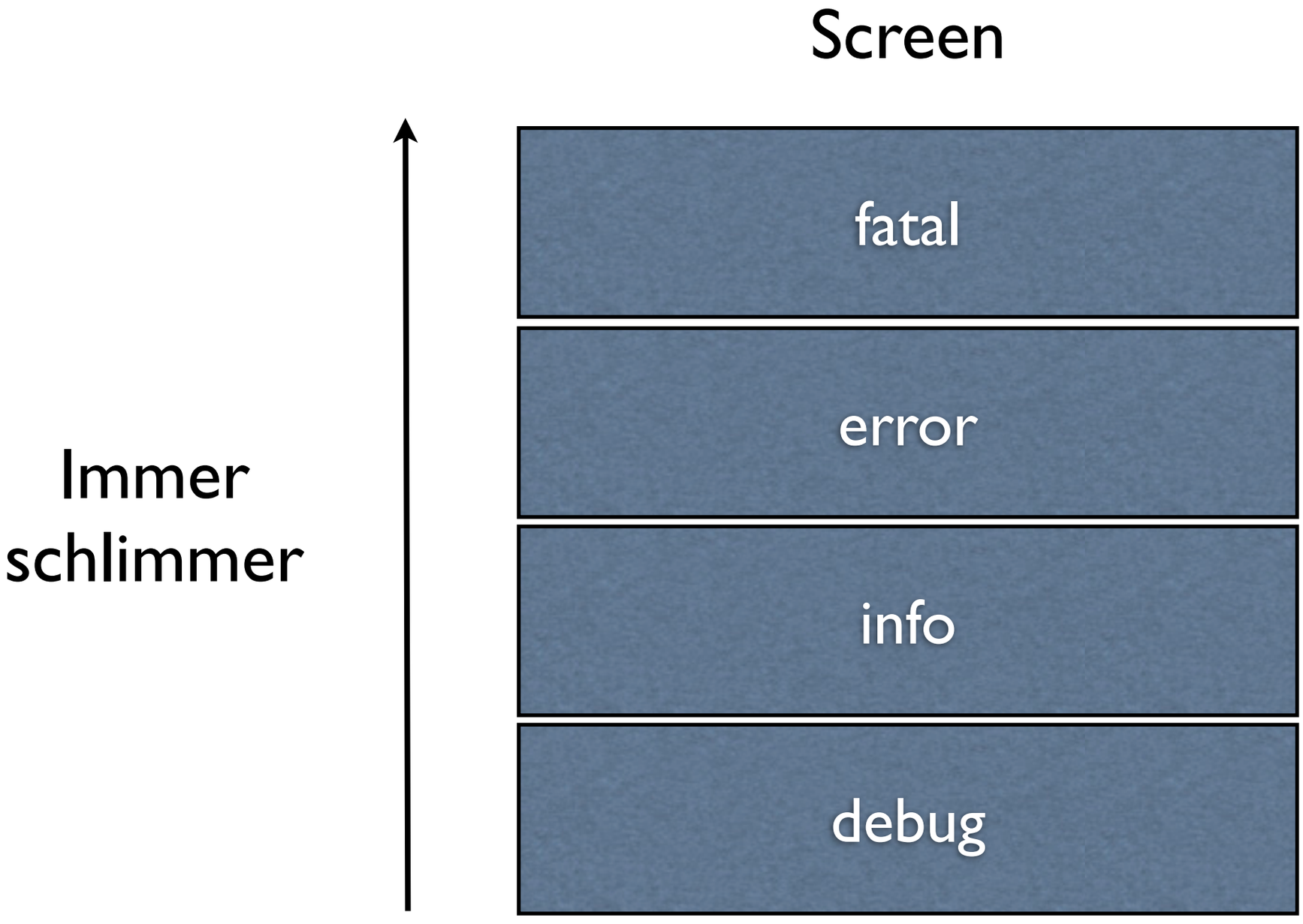
Mehr Probleme...

```
# Logging starten
use Log::Log4perl qw/:easy/;
Log::Log4perl->easy_init($ERROR);
my $logger = Log::Log4perl->get_logger;

$logger->debug("$0 ist gestartet");
...
$logger->info("Programmuser ist: $user");

$fk = Firma::Kunden->new("...");
$alle_kunden = $fk->hole_kundenliste;
$logger->error(sprintf("Nur %d Kunden?", $alle_kunden->anzahl))
    if $alle_kunden->anzahl < 100;
$angeb_kunde = $fk->hole_angebotsdaten(kunde => 'oberdorf');
$logger->logdie("Angebotsdaten nicht bekommen")
    unless defined $angeb_kunde;
$logger->debug("Habe alle Daten für das Angebot");
...
$ausgabe = Firma::Ausgabe->new(
    kunde => $angeb_kunde,
    kundenliste => $alle_kunden,
    ausgabeart => ... );
```

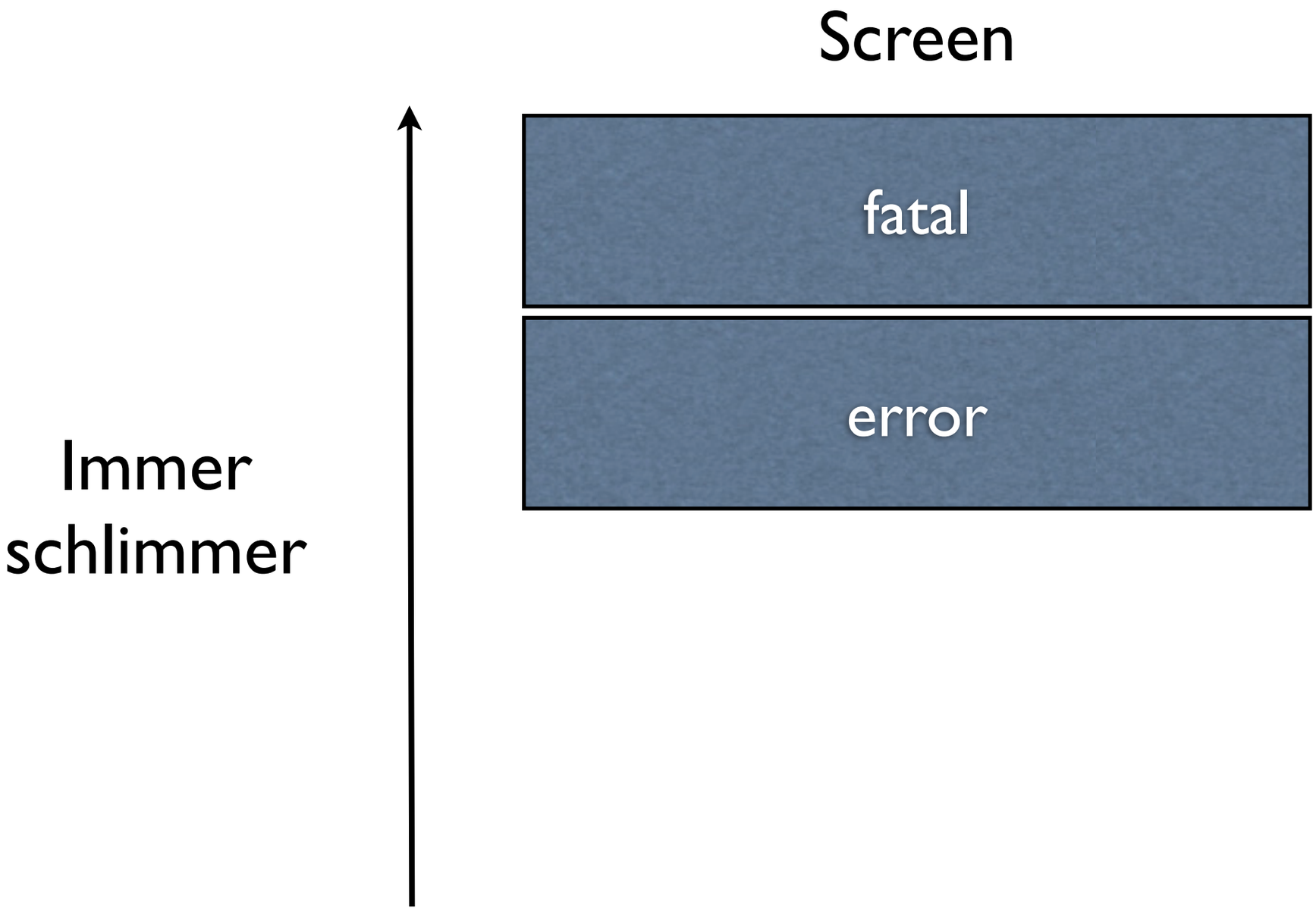
Screen



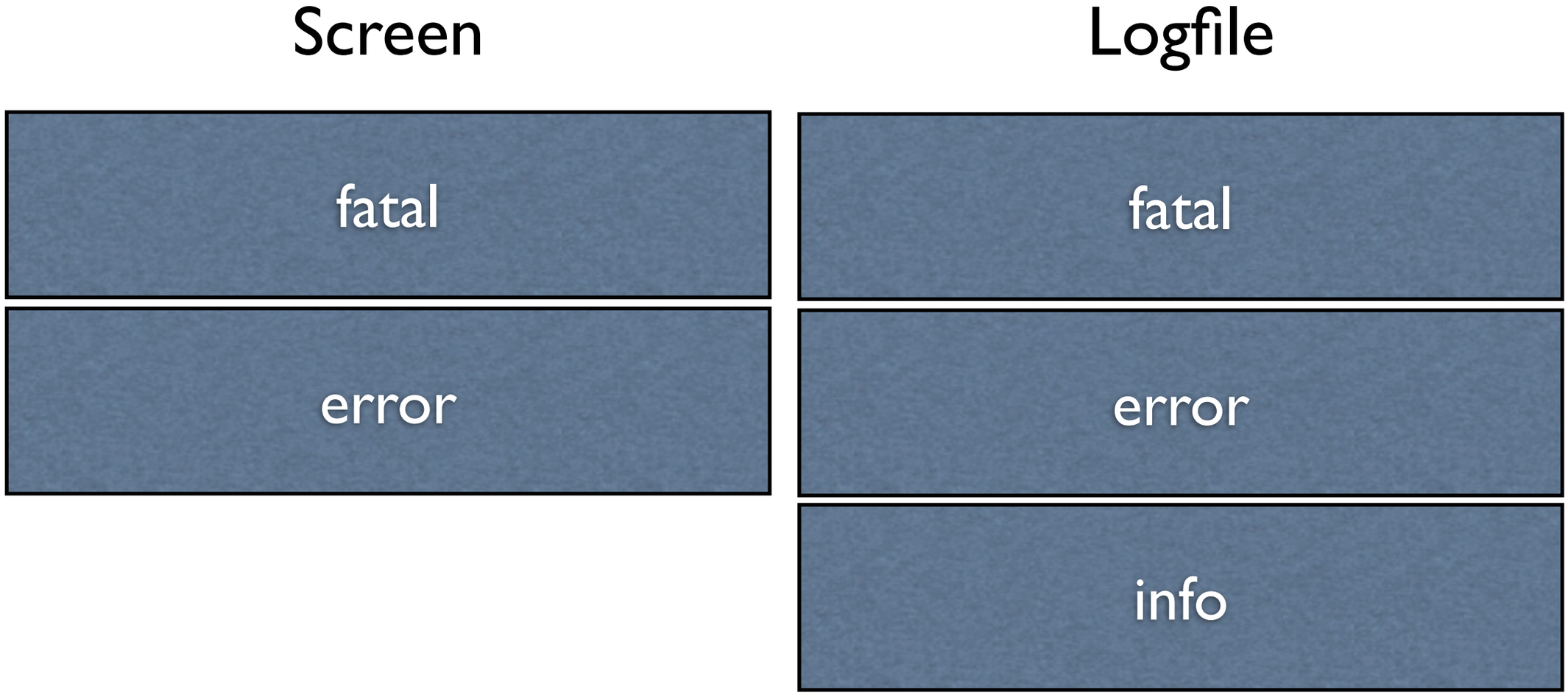
Die Loglevel



Loglevel beim Entwickeln



Loglevel im Einsatz



Feinere Loglevel im Einsatz

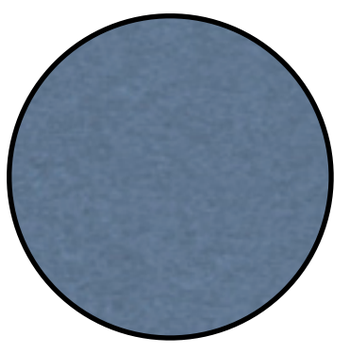
Zusätzlich in eine Datei loggen...

```
use Log::Log4perl;

my $log_conf = q{
    log4perl.category = INFO, Logfile, Screen

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = angebot.log
    log4perl.appender.Logfile.mode = append

    log4perl.appender.Screen = Log::Log4perl::Appender::Screen
};

Log::Log4perl::init( \$log_conf );
my $logger = Log::Log4perl::get_logger();
```

Ich will doch nur...

... aber das Projekt wird immer größer!

angebot.pl
--kunde=zdf



Firma::Kunden

http

XML

angebot.pl
--kunde=zdf

Kundenadr.
rausfinden

Firma::Preise

Preisanfrage
für Kunde

Datenbank

$output_fn
$var = {...}

Firma::Angebot

Doku
ment

Loggen nervt

- Der Brennpunkt der Arbeit liegt jeden Tag in anderen Bereichen des Programmes
- Das Hochsetzen der Logausgaben verdirbt meinen Bildschirm
- Wichtige Dinge können nicht mehr erkannt werden

Gezieltes Verändern der Loglevel für Module

```
use Log::Log4perl;

my $log_conf = q{
    log4perl.category = INFO, Logfile, Screen

    log4perl.logger.main = ERROR
    log4perl.logger.Firma.Kunden = ERROR
    log4perl.logger.Firma.Preise = DEBUG
    log4perl.logger.Firma.Angebot = ERROR

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = angebot.log
    log4perl.appender.Logfile.mode = append

    log4perl.appender.Screen = Log::Log4perl::Appender::Screen
};

Log::Log4perl::init( \$log_conf );
my $logger = Log::Log4perl::get_logger('main');
```

Loglevel für Logbereiche (oft Module, aber nicht zwingend)

```
package Firma::Preise;

sub new {
    my($class,%args) = @_;
    my $self = bless { %args }, $class;
    my $logger = $self->_get_logger;
    # Prüfe Argumente
    ...
    $logger->debug("Argumente alle ok");
    ...
}
```

```
sub _get_logger {
    Log::Log4perl::get_logger('Firma.Preise');
}
```

Loggen nervt wenig mit Log::Log4perl

- Einfacher Einstieg
- Flexibles Logging durch Inistring
- Flexible Logwege: Screen, File, Datenbank, XML, Syslog, NT-Eventlog ...
- Loggen mit Timestamp, Modulname, Funktionsname möglich durch Inistring
- Verschiedene Loglevel für kalte Stellen und Brennpunkte im Programm
- Loglevel in Brennpunkten hoch und wieder runtersetzen $logger->level ($DEBUG)

Loggen nervt wenig mit Log::Log4perl

- Verfügbar für Linux
- Verfügbar für Activestate Perl/Win als PPM
- Wechsel der Loglevels im Programmlauf durch Verändern einer „Inidatei“ für Log4perl möglich

Log::Log4perl

Einfach loggen...

Danke...

Richard Lippmann
horshack@lisa.franken.de