

# Log::Log4perl

Logging the simple way...

Richard Lippmann  
horshack@lisa.franken.de

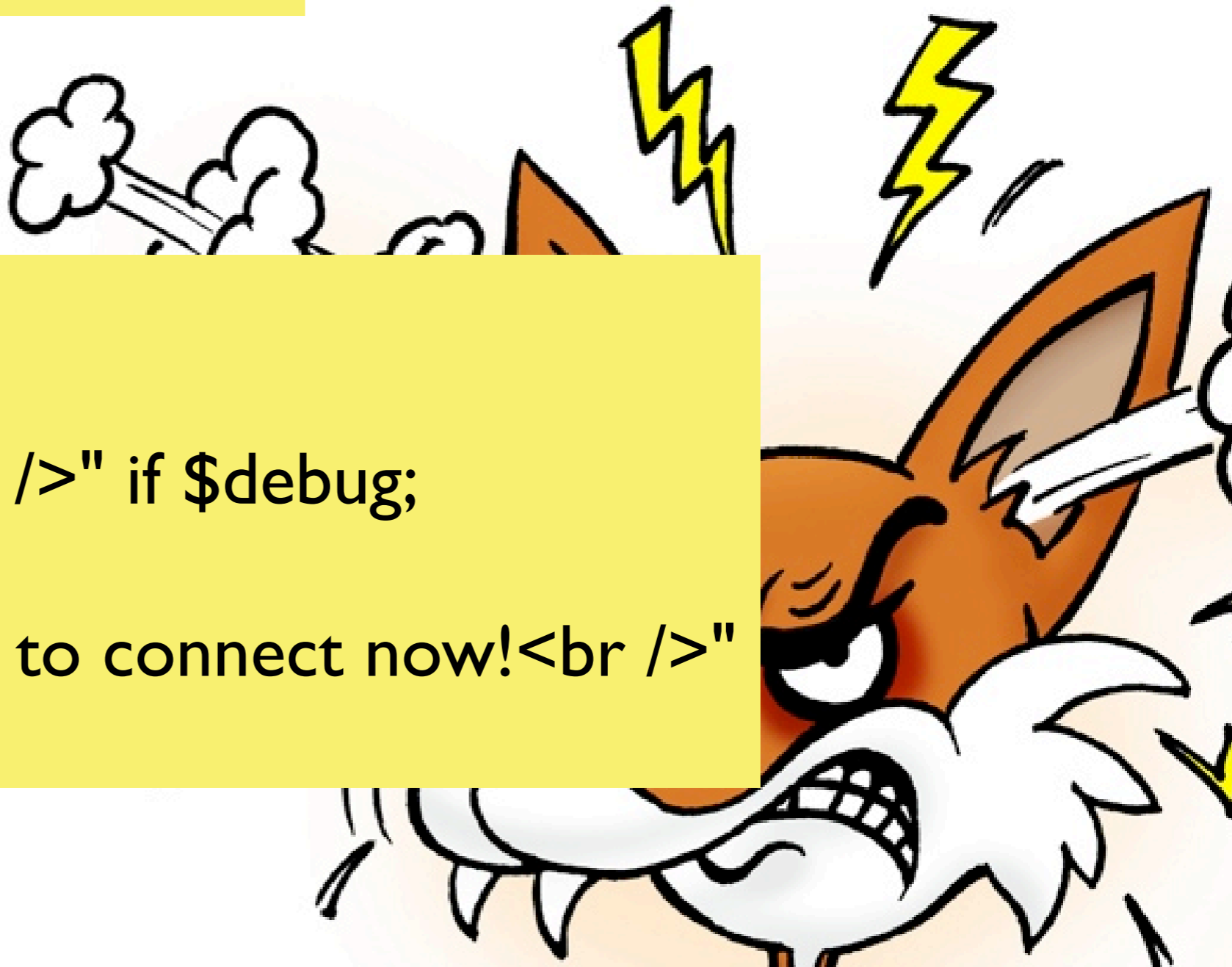


Logging ist totally easy:

```
our $debug=1;  
...  
print "open database\n" if $debug;  
...
```

... or sometimes:

```
our $debug=1;  
...  
print "open database<br />" if $debug;  
...  
print "database lives? try to connect now!<br />"  
  if $debug;
```



... in a module:

```
...  
print "open database<br />" if $main::debug;  
...  
print "database lives? try to connect now!<br />"  
  if $debug; # Ooops! Variable not declared here
```

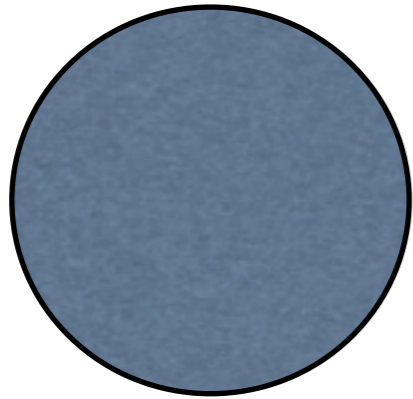


# Logging is annoying



- I have to write a program and don't have time to take care about logging
- Logging spoils my screen
- Important issues cannot be recognized
- In the end all print-commands must get removed

offer.pl  
--customer=zdf



I only want to...

1. Everything started with this sentence
2. How much time do we want to waste to re-invent everything again?
3. Do we have this time?



A little start. Everything works, so we don't need more than this.

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

$logger->debug("$0 has been started");

...
$fk = Company::Customers->new("...");
@all_customers = $fk->get_list_of_customers;
$customer_offer = $fk->get_data_for_offer(customer => 'london');

...
$output = Company::Output->new(
    customer => $customer_offer,
    list_of_customers => \@all_customers,
    type_of_output => ... );
```



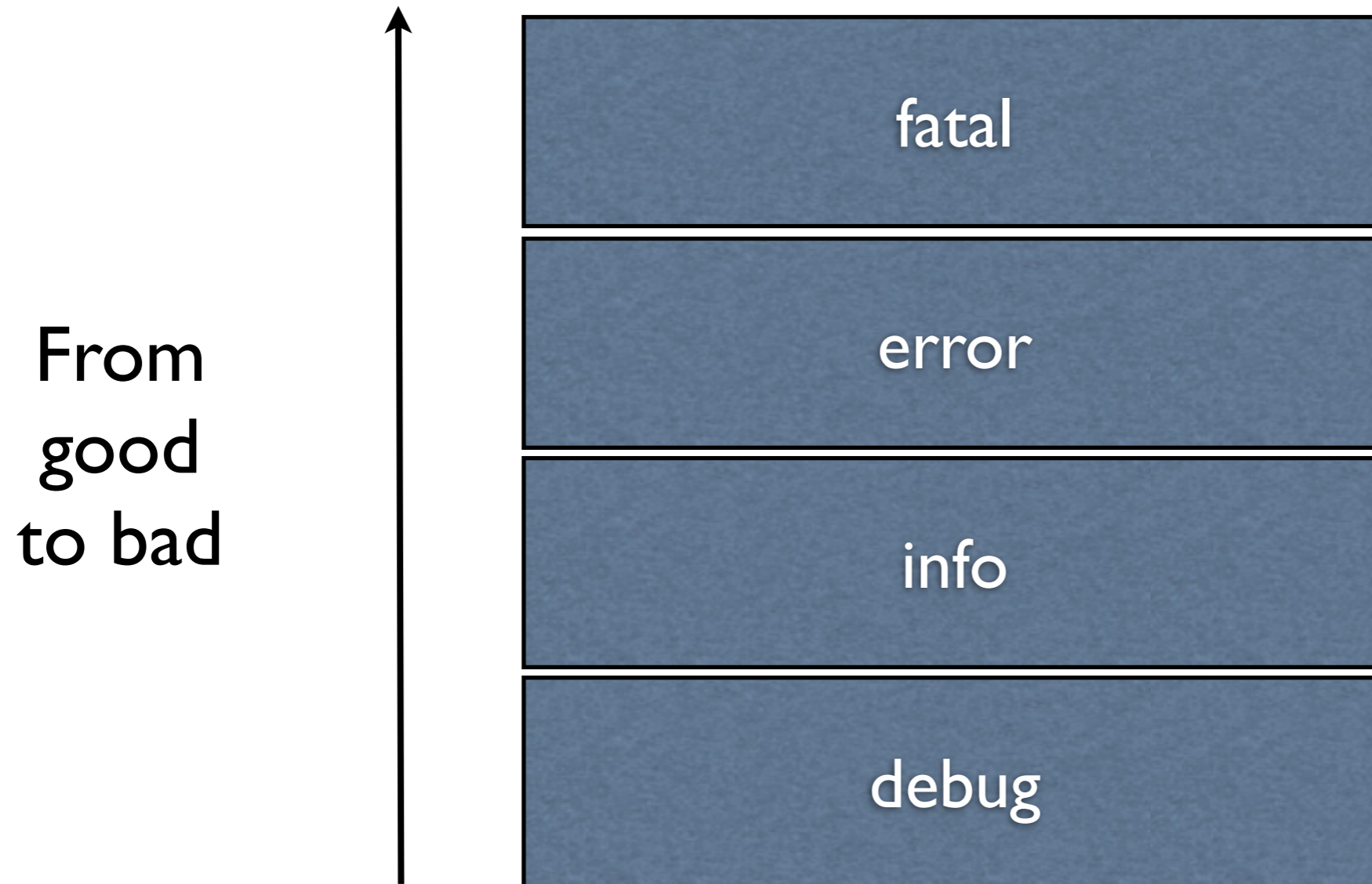
# First problems arise...

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

$logger->debug("$0 has been started");
...
$logger->info("user of program is: $user");
$fk = Company::Customers->new("...");
@all_customers = $fk->get_list_of_customers;
$logger->error(sprintf("Only %d customers?", $fk->amount))
    if $fk->amount < 100;
$customer_offer = $fk->get_data_for_offer(customer => 'london');
$logger->logdie("Didn't get offer-data")
    unless defined $customer_offer;
$logger->debug('Did get data for offer successfully');
...
$output = Company::Output->new(
    customer => $customer_offer,
    list_of_customers => \@all_customers,
    type_of_output => ... );
...
```

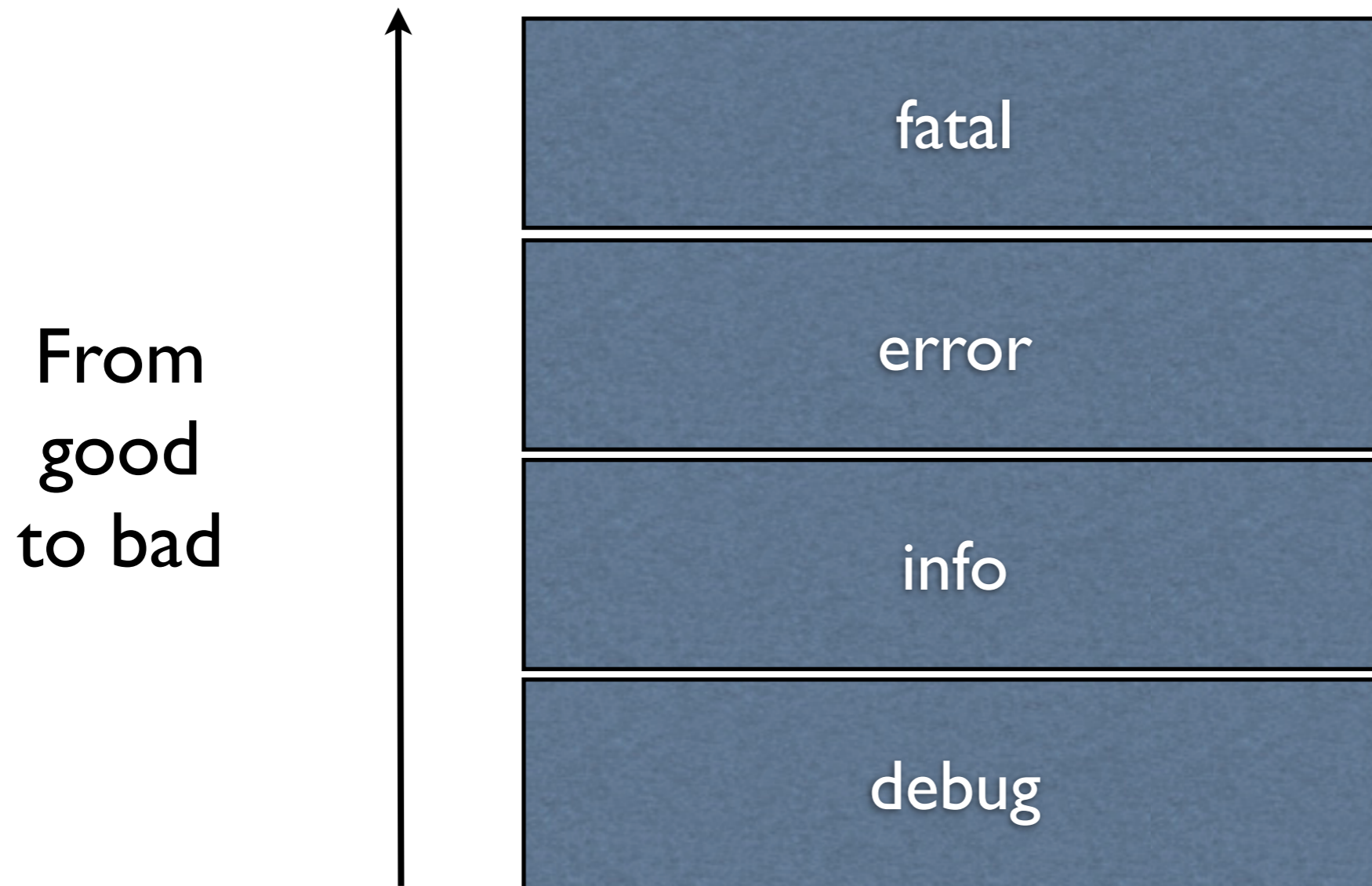


# Screen



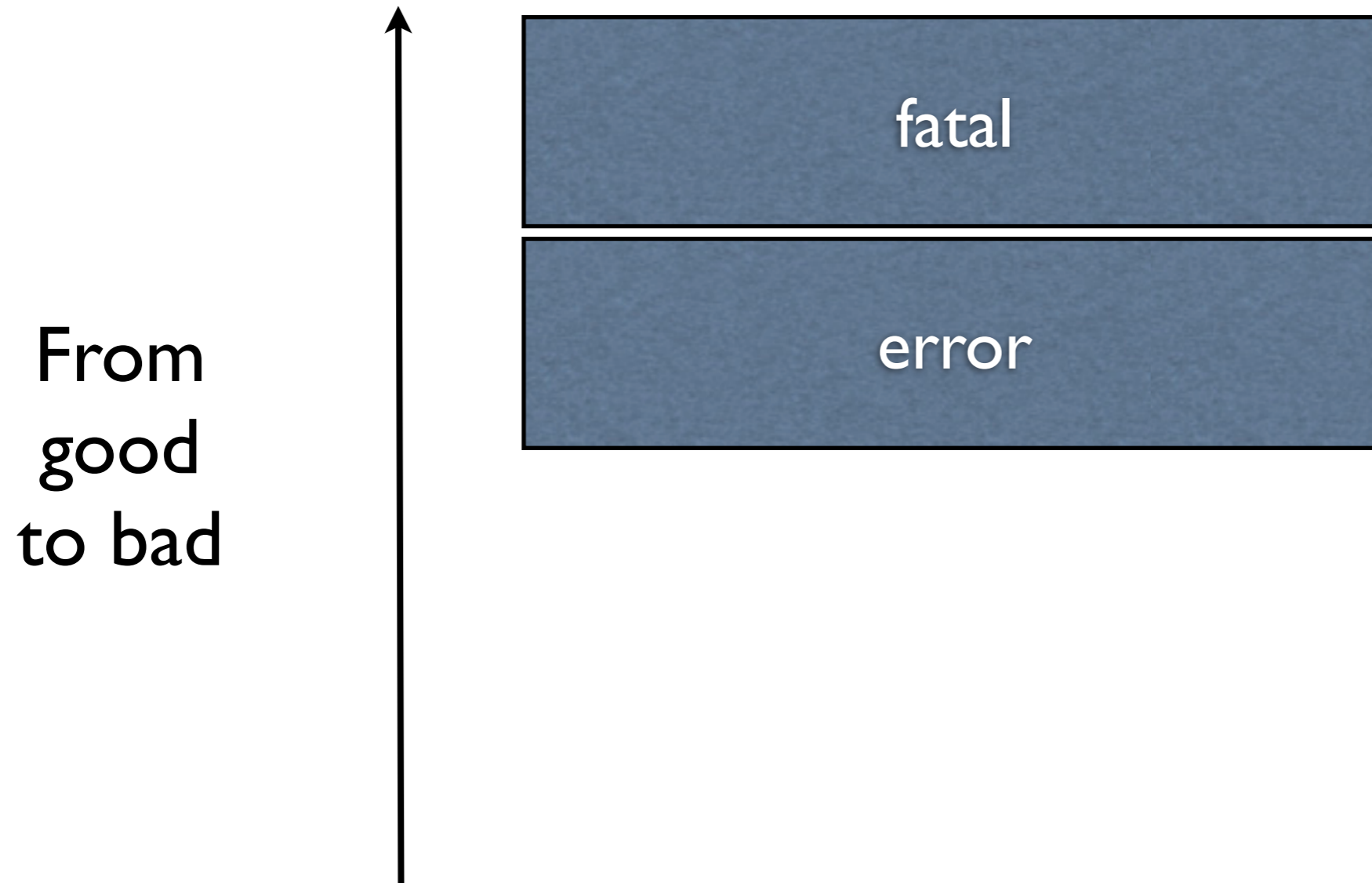
## levels of logging

# Screen



levels of logging when developing

# Screen



# levels of logging in production

Screen

Logfile

fatal

fatal

error

error

info

separated

levels of logging in production

# Log additionally to a file...

```
use Log::Log4perl;
```

```
my $log_conf = q{  
    log4perl.category = INFO, Logfile, Screen  
  
    log4perl.appender.Logfile = Log::Log4perl::Appender::File  
    log4perl.appender.Logfile.filename = offer.log  
    log4perl.appender.Logfile.mode = append  
  
    log4perl.appender.Screen = Log::Log4perl::Appender::Screen  
};  
  
Log::Log4perl::init( \ $log_conf );  
my $logger = Log::Log4perl::get_logger();
```

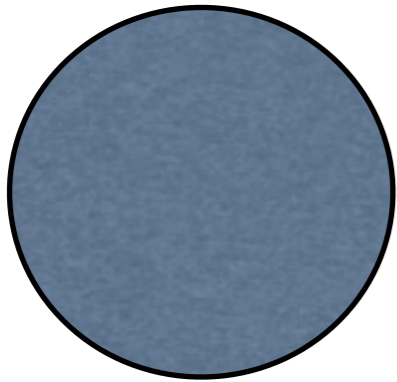


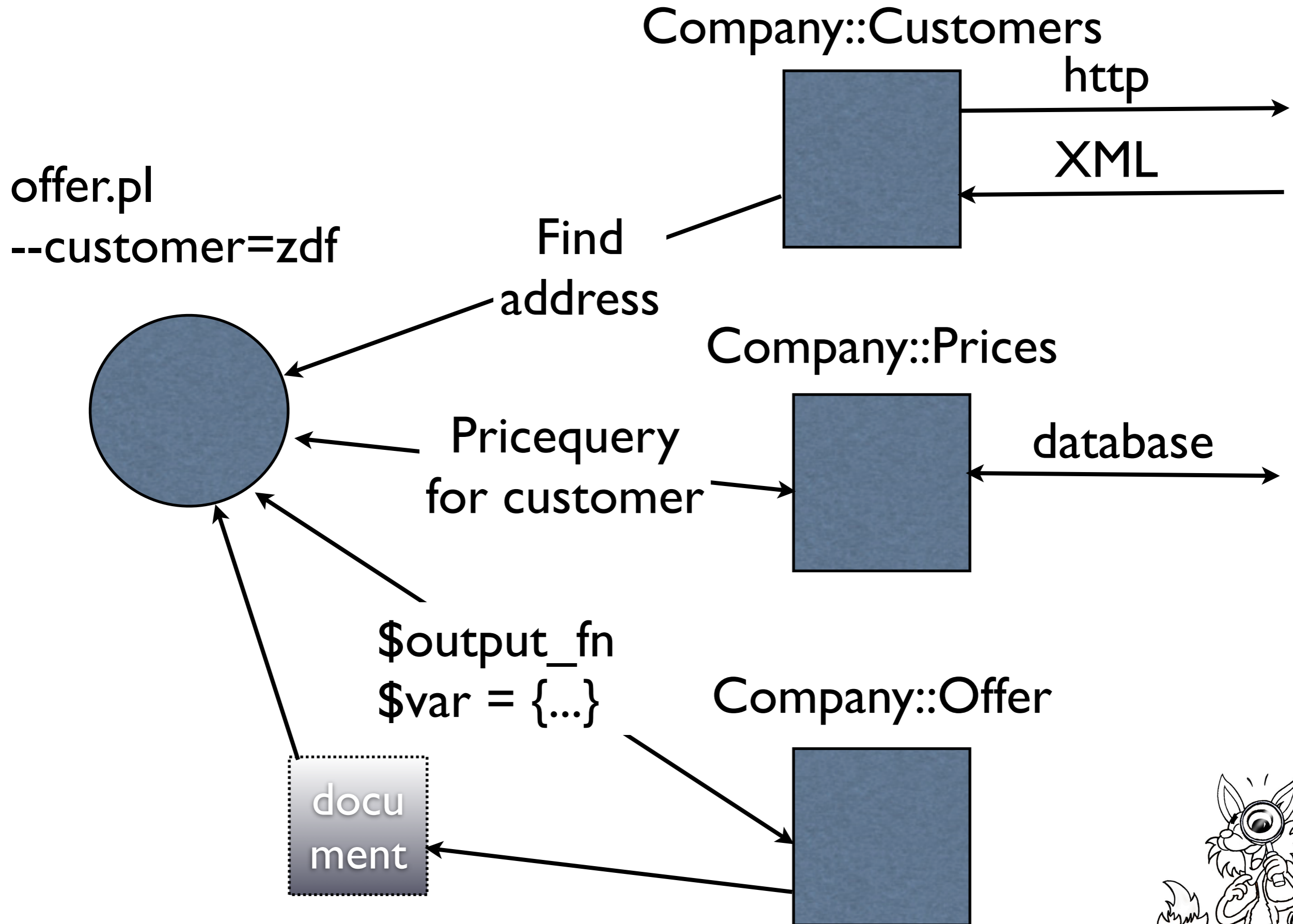
I only want to...

... but the project is getting larger!

offer.pl

--customer=zdf





# Logging is annoying



- I have to write a program and don't have time to take care about logging
- Logging spoils my screen
- Important issues cannot be recognized
- In the end all print-commands must get removed

# Selective changing of loglevels for modules

```
use Log::Log4perl;
```

```
my $log_conf = q{
```

```
    log4perl.category = INFO, Logfile, Screen
```

```
    log4perl.logger.main = ERROR
```

```
    log4perl.logger.Company.Customers = ERROR
```

```
    log4perl.logger.Company.Prices = DEBUG
```

```
    log4perl.logger.Company.Offer = ERROR
```

```
    log4perl.appender.Logfile = Log::Log4perl::Appender::File
```

```
    log4perl.appender.Logfile.filename = offer.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');
```



# Selective loglevels for different parts of the program (often modules, but not mandatory)

```
package Company::Prices;

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

```
sub _get_logger {
    Log::Log4perl::get_logger('Company.Prices');
}
```



# Logging annoys less with Log::Log4perl

- Simple start
- Flexible Logging through Inistring
- Flexible ways of logging: screen, file, database, XML, syslog, NT-eventlog ...
- Logging with timestamp, modulename, name of function possible through ini-string
- different loglevels for cold spots and hot spots in your program
- Change loglevels in the code `$logger->level ($DEBUG)`
- Set loglevel with external file read on runtime



# Logging annoys less with Log::Log4perl

- Available für Linux
- Available for Activestate Perl/Windows as PPM



# Log::Log4perl

Some examples if time allows...

Richard Lippmann  
horshack@lisa.franken.de



# Log::Log4perl

A fast approach, log to screen

Richard Lippmann  
horshack@lisa.franken.de



# A fast approach, log to screen

When you simply want to log to screen  
from a certain level on:

```
use Log::Log4perl qw(:easy);  
# Set priority of root logger to ERROR  
Log::Log4perl->easy_init($ERROR);  
  
# Application Section  
my $logger = get_logger();  
# Will not be logged:  
$logger->info("I do this and that");  
# Will be logged:  
$logger->error("Bad");
```

Logging looks like this:

```
~horshack$ perl test.pl  
2006/06/18 02:48:13 Bad
```



# Log::Log4perl

Log to screen and file

Richard Lippmann  
horshack@lisa.franken.de



A string, a simple string!

```
use Log::Log4perl;

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

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = your_logfilename.log
    log4perl.appender.Logfile.mode = write
    log4perl.appender.Logfile.layout = Log::Log4perl::Layout::SimpleLayout

    log4perl.appender.Screen          = Log::Log4perl::Appender::Screen
    log4perl.appender.Screen.layout = Log::Log4perl::Layout::SimpleLayout
/;

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

$logger->info("Starting $0");
$logger->error("Bad thing happened");
```



# Log::Log4perl

Different loglevels for hot spots in the program

Richard Lippmann  
horshack@lisa.franken.de



# Different loglevels for hot spots in the program

```
# Logging (mode=write|append)
my $log_conf = q/
    log4perl.category = INFO, Logfile

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = your_logfilename.log
    # mode can be write|append
    log4perl.appender.Logfile.mode = write
    log4perl.appender.Logfile.layout = Log::Log4perl::Layout::PatternLayout
    log4perl.appender.Logfile.layout.ConversionPattern = %d %p> %m%n

    log4perl.appender.Screen = Log::Log4perl::Appender::Screen
    log4perl.appender.Screen.stderr = 0
    log4perl.appender.Screen.layout = Log::Log4perl::Layout::PatternLayout
    log4perl.appender.Screen.layout.ConversionPattern = %d %p> %m<br />%n

    log4perl.logger.main = INFO
    log4perl.logger.Hob.Inventory = INFO
    log4perl.logger.Hob.GetWishlist = INFO
    log4perl.logger.Hob.Harvester = INFO
    log4perl.logger.Hob.Harvester.Anna = INFO
    log4perl.logger.Hob.Harvester.Petra = DEBUG
    log4perl.logger.Hob.Harvester.Ludmilla = INFO
/;

# Create Singleton-Object
Log::Log4perl::init( \$log_conf );
# Logging ready

my $logger = Log::Log4perl::get_logger("main");
$log->info("Starting $0");
# later
my $logger = Log::Log4perl::get_logger("Hob.Harvester.Petra");
$log->info("Entering Harvester::Petra now");
```



# Different loglevels for hot spots in the program

```
# Logging (mode=write|append)
my $log_conf = q/
  log4perl.category = INFO, Logfile

  log4perl.appender.Logfile = Log::Log4perl::Appender::File
  log4perl.appender.Logfile.filename = your_logfilename.log
  # mode can be write|append
  log4perl.appender.Logfile.mode = write
  log4perl.appender.Logfile.layout = Log::Log4perl::Layout::PatternLayout
  log4perl.appender.Logfile.layout.ConversionPattern = %d %p> %m%n

  log4perl.appender.Screen = Log::Log4perl::Appender::Screen
  log4perl.appender.Screen.stderr = 0
  log4perl.appender.Screen.layout = Log::Log4perl::Layout::PatternLayout
  log4perl.appender.Screen.layout.ConversionPattern = %d %p> %m<br />%n

  log4perl.logger.main = INFO
  log4perl.logger.Hob.Inventory = INFO
  log4perl.logger.Hob.GetWishlist = INFO
  log4perl.logger.Hob.Harvester = INFO
  log4perl.logger.Hob.Harvester.Anna = INFO
  log4perl.logger.Hob.Harvester.Petra = DEBUG
  log4perl.logger.Hob.Harvester.Ludmilla = INFO
/;
# Create Singleton-Object
Log::Log4perl::init( \$log_conf)
# Logging ready

my $logger = Log::Log4perl::get_logger("main");
$log->info("Starting $0");
```



2006/06/10 12:14:22 INFO> Login started



# Different loglevels for hot spots in the program

```
# Logging (mode=write|append)
my $log_conf = q/
  log4perl.category = INFO, Logfile

  log4perl.appender.Logfile = Log::Log4perl::Appender::File
  log4perl.appender.Logfile.filename = your_logfilename.log
  # mode can be write|append
  log4perl.appender.Logfile.mode = write
  log4perl.appender.Logfile.layout = Log::Log4perl::Layout::PatternLayout
  log4perl.appender.Logfile.layout.ConversionPattern = %d %p> %m%n

  log4perl.appender.Screen = Log::Log4perl::Appender::Screen
  log4perl.appender.Screen.stderr = 0
  log4perl.appender.Screen.layout = Log::Log4perl::Layout::PatternLayout
  log4perl.appender.Screen.layout.ConversionPattern = %d %p> %m<br />%n

  log4perl.logger.main = INFO
  log4perl.logger.Hob.Inventory = INFO
  log4perl.logger.Hob.GetWishlist = INFO
  log4perl.logger.Hob.Harvester = INFO
  log4perl.logger.Hob.Harvester.Anna = INFO
  log4perl.logger.Hob.Harvester.Petra = DEBUG
  log4perl.logger.Hob.Harvester.Ludmilla = INFO
/;
# Create Singleton-Object
Log::Log4perl::init( \$log_conf );
# Logging ready

my $logger = Log::Log4perl::get_logger("main");
$log->info("Starting $0");
```

not needed now



# Different loglevels for hot spots in the program

```
# Logging (mode=write|append)
my $log_conf = q/
    log4perl.category = INFO, Logfile

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = your_logfilename.log
    # mode can be write|append
    log4perl.appender.Logfile.mode = write
    log4perl.appender.Logfile.layout = Log::Log4perl::Layout::PatternLayout
    log4perl.appender.Logfile.layout.ConversionPattern = %d %p> %m%n

    log4perl.appender.Screen = Log::Log4perl::Appender::Screen
    log4perl.appender.Screen.stderr = 0
    log4perl.appender.Screen.layout = Log::Log4perl::Layout::PatternLayout
    log4perl.appender.Screen.layout.ConversionPattern = %d %p> %m<br />%n

    log4perl.logger.main = INFO
    log4perl.logger.Hob.Inventory = INFO
    log4perl.logger.Hob.GetWishlist = INFO
    log4perl.logger.Hob.Harvester = INFO
    log4perl.logger.Hob.Harvester.Annna = INFO
    log4perl.logger.Hob.Harvester.Petra = DEBUG
    log4perl.logger.Hob.Harvester.Ludmilla = INFO
/;

# Create Singleton-Object
Log::Log4perl::init( \$log_conf );
# Logging ready

my $logger = Log::Log4perl::get_logger("main");
$log->info("Starting $0");
```

Trouble happens here!



# Log::Log4perl

Determine name of logfile on runtime

Richard Lippmann  
horshack@lisa.franken.de



# Determine name of logfile on runtime

```
# Logging (mode=write|append)
my $log_conf = q/
    log4perl.category = INFO, Logfile

    log4perl.appender.Logfile = Log::Log4perl::Appender::File
    log4perl.appender.Logfile.filename = sub { return get_log_fn(); }
    # mode can be write|append
    log4perl.appender.Logfile.mode = write
    log4perl.appender.Logfile.layout = Log::Log4perl::Layout::PatternLayout
    log4perl.appender.Logfile.layout.ConversionPattern = %d %p> %m%n

...
    log4perl.logger.main = INFO
    log4perl.logger.Hob.Inventory = INFO
    log4perl.logger.Hob.GetWishlist = INFO
/;
# Create Singleton-Object
Log::Log4perl::init( \$log_conf );
# Logging ready

my $logger = Log::Log4perl::get_logger("main");
$log->info("Starting $0");

=head2 get_log_fn

Return Logfilename for this programm. It is the basename of this programm
with a ".log" ending.

=cut

sub get_log_fn {
    use File::Basename;
    return sprintf "%s.log", basename( $0, '.pl' );
}
```

use a subroutine



# Log::Log4perl

Use an external configfile for logging

Richard Lippmann  
horshack@lisa.franken.de



# Use an external configfile for logging

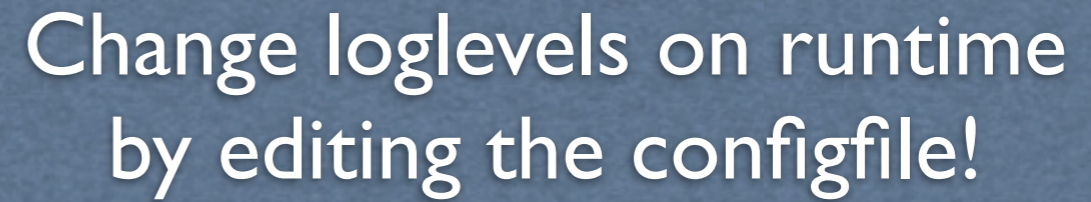
simply init from logfile

```
use Log::Log4perl;  
  
my $conf_file = 't.logconfig';  
# read config file in the beginning  
Log::Log4perl->init( $conf_file );  
  
my $logger = Log::Log4perl::get_logger('main');  
$logger->debug("Starting $0");
```



# Use an external configfile for logging

Change loglevels on runtime  
by editing the configfile!



```
use Log::Log4perl;

my $conf_file = 't.logconfig';
# Re-read config file every 60 seconds
Log::Log4perl->init_and_watch( $conf_file, 60 );

my $logger = Log::Log4perl::get_logger('main');
$logger->debug("Starting $0");
```



# Log::Log4perl

Use a section in your inifile of your application

Richard Lippmann  
horshack@lisa.franken.de



# Use a section in your inifile of your application

This looks exactly like the section of a logfile!

```
log4perl.category = INFO, Logfile, Screen

log4perl.appender.Logfile = Log::Log4perl::Appender::File
log4perl.appender.Logfile.filename = your_logfile.log
log4perl.appender.Logfile.mode = write

log4perl.appender.Logfile.layout = Log::Log4perl::Layout::SimpleLayout
log4perl.appender.Screen = Log::Log4perl::Appender::Screen
log4perl.appender.Screen.layout = Log::Log4perl::Layout::SimpleLayout
```



# Use a section in your inifile of your application

Put it into your inifile

```
[database]
dsn=DBI:mysql:database=flowers;host=mysql.local
user=me
```

## **[log4perl]**

```
log4perl.category = INFO, Logfile, Screen
```

```
log4perl.appender.Logfile = Log::Log4perl::Appender::File
log4perl.appender.Logfile.filename = your_logfile.log
log4perl.appender.Logfile.mode = write
```

```
log4perl.appender.Logfile.layout = Log::Log4perl::Layout::SimpleLayout
log4perl.appender.Screen = Log::Log4perl::Appender::Screen
log4perl.appender.Screen.layout = Log::Log4perl::Layout::SimpleLayout
```



# Use a section in your inifile of your application

Read it and initialize log4perl  
with it

```
my $ini_obj = Config::IniFiles->new( -file => 'my.ini');

my $log_conf;
foreach my $var ( $ini_obj->Parameters('log4perl') ) {
    next unless defined $var;
    $log_conf .= sprintf "%s=%s\n",
        $var, $ini_obj->val( 'log4perl', $var );
}
Log::Log4perl::init( \$log_conf );
```



# Log::Log4perl

## Logrotate

Richard Lippmann  
horshack@lisa.franken.de



Module FileRotate shall care  
about it



```
...  
log4perl.appender.Logfile = Log::Dispatch::FileRotate  
log4perl.appender.Logfile.filename = findus.log  
log4perl.appender.Logfile.mode = append  
log4perl.appender.Logfile.max = 2  
log4perl.appender.Logfile.size = 10_000_000  
...
```



# Log::Log4perl

Logging the simple way...

**Thanks for listening!**

Richard Lippmann  
horshack@lisa.franken.de

