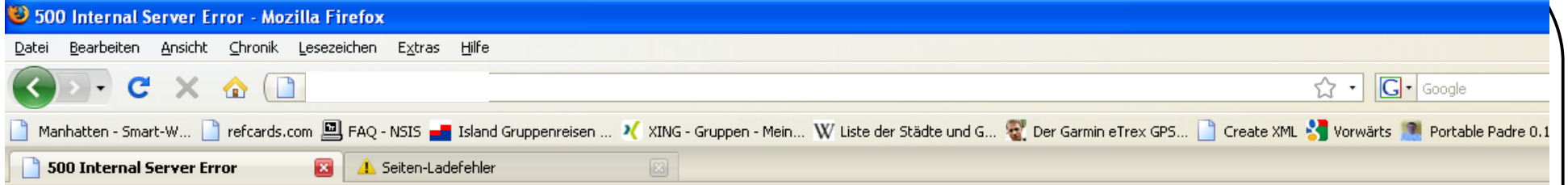


Module für nützliche Infos



Internal Server Error

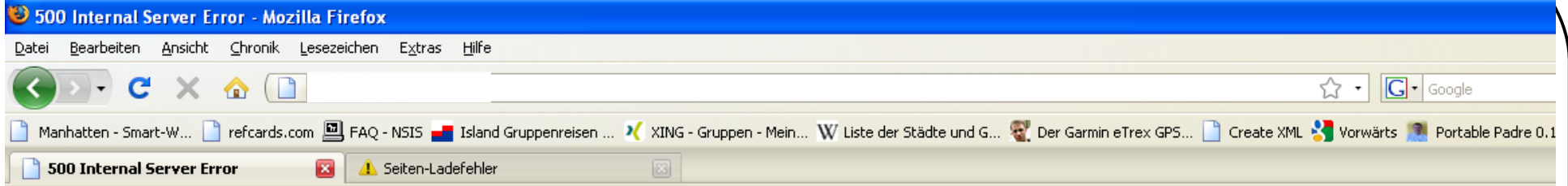
The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, webmaster@computer.net and inform them of the time the error occurred, and anything you might have done that may have caused the error.

More information about this error may be available in the server error log.

Apache/1.3.37 Server at 127.0.0.1 Port 80

Fertig



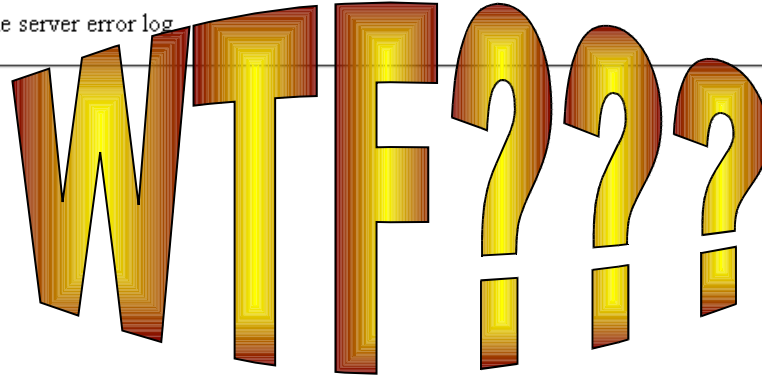
Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, webmaster@computer.net and inform them of the time the error occurred, and anything you might have done that may have caused the error.

More information about this error may be available in the server error log

Apache/1.3.37 Server at 127.0.0.1 Port 80



Module für nützliche Infos

- Nicht nur für Webentwicklung
- PadWalker
- Datenbank-Logging
- „Interna“
- <http://renee-baecker.de/vortraege.html>

\$foo - Perl-Magazin (Workshops, Module, News und vieles mehr) - Mozilla Firefox

[Datei](#) [Bearbeiten](#) [Ansicht](#) [Chronik](#) [Lesezeichen](#) [Extras](#) [Hilfe](#)

[http://foo-magazin.de/order.cgi](#)

[Manhattan - Smart-W...](#) [refcards.com](#) [FAQ - NSIS](#) [Island Gruppenreisen ...](#) [XING - Gruppen - Mein...](#) [W Liste der Städte und G...](#) [Der Garmin eTrex GPS...](#) [Create XML](#) [Vorwärts](#) [Porta](#)

[Perl-Community.de: Thread: Perl-Tk un...](#) [Google Reader \(1\)](#) [XING - Gruppen - Meine Gruppen - Suc...](#) [Confirmation email sent](#) [\\$foo - Perl](#)



// SEIBERT / MEDIA
PERL-PROGRAMMIERER/IN GESUCH
 FÜR E-BUSINESS APPLICATIONS
[HIER KLICKEN UND ONLINE BEWERBEN!](#)

Ausgabe
Inhalt
Leseprobe
Beispiele herunterladen
Archiv
Bestellung
\$foo bestellen
Mein Warenkorb
Abonnenten
Service
Downloads
Newsletter
Leserbrief
Preise
Datenschutz
Mediadaten

\$foo - Bestellung -> Adresse

Adresse

Bitte geben Sie die Adresse ein. Sollte die Rechnungsadresse von der Lieferadresse abweichen, bitte hier die Rechnungsadresse eintragen und danach über den Button "extra Lieferadresse angeben" eine Lieferadresse angeben. **Alle Felder mit * müssen ausgefüllt werden!**

Kundennummer (falls vorhanden)

Firma (für Firmenkunden)

Anrede *

Vorname * Name *

Straße / Hnr. *

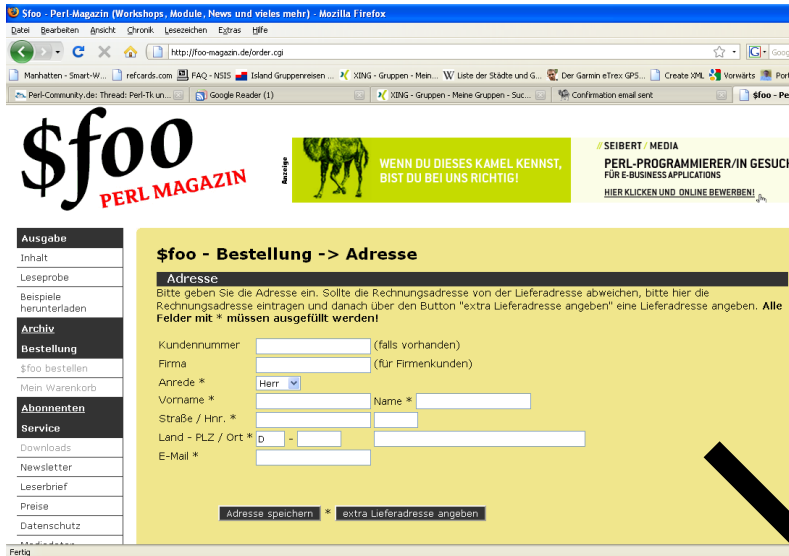
Land - PLZ / Ort * -

E-Mail *

*

Fertig

PadWalker



Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, webmaster@computer.net and inform them of the time the error occurred, a

More information about this error may be available in the server error log.

Apache/1.3.37 Server at 127.0.0.1 Port 80

PadWalker



Was ist passiert?
Was hat der User eingegeben?
Welche Werte hatten andere Variablen?
Viele Fragen, die zum Reproduzieren des Fehlers beantwortet werden müssen

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, webmaster@computer.net and inform them of the time the error occurred, a

More information about this error may be available in the server error log.

Apache/1.3.37 Server at 127.0.0.1 Port 80

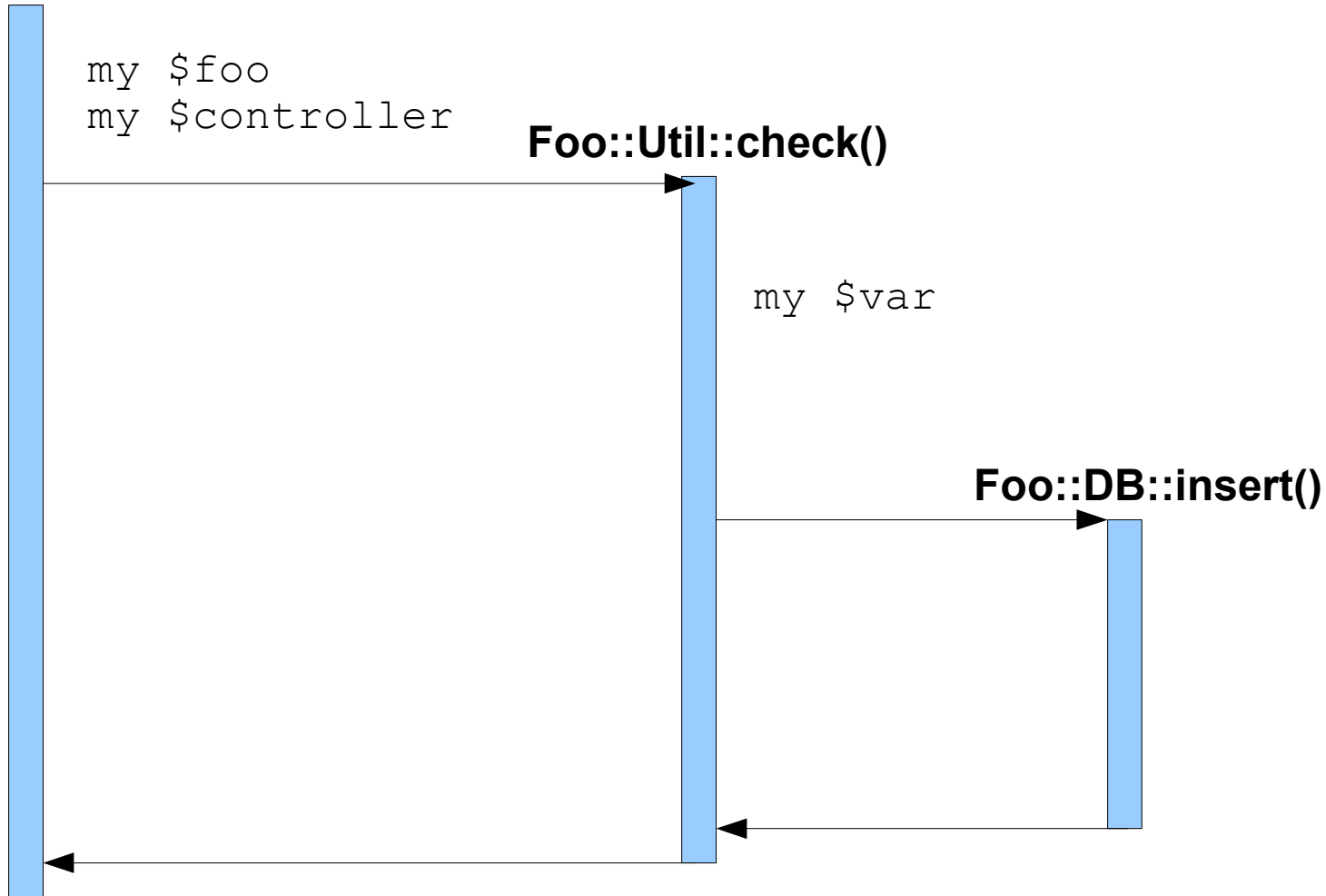
PadWalker
to the
rescue

PadWalker

- Nützlich, um z.B. Eingabedaten zu bekommen und per Mail an Webmaster zu schicken
 - Ermöglicht das Testen anhand von „Realdaten“
 - Mit DIE-Signalhandler verbinden
 - Mit Logging verbinden

PadWalker

Foo::Main::main()



PadWalker

Foo::Main::main()

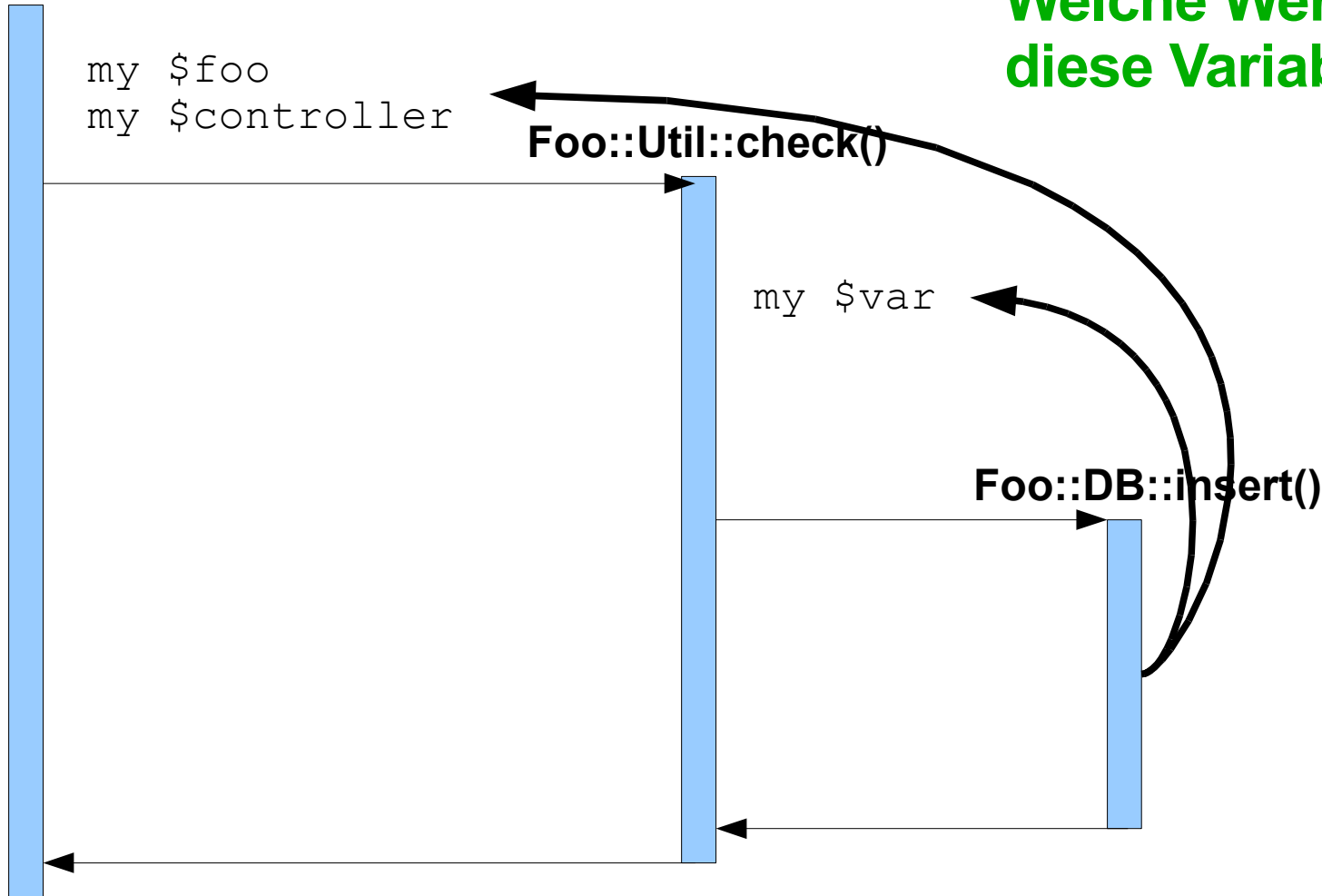
my \$foo
my \$controller

Foo::Util::check()

my \$var

Foo::DB::insert()

Welche Werte haben diese Variablen??



PadWalker

Foo::Main::main()

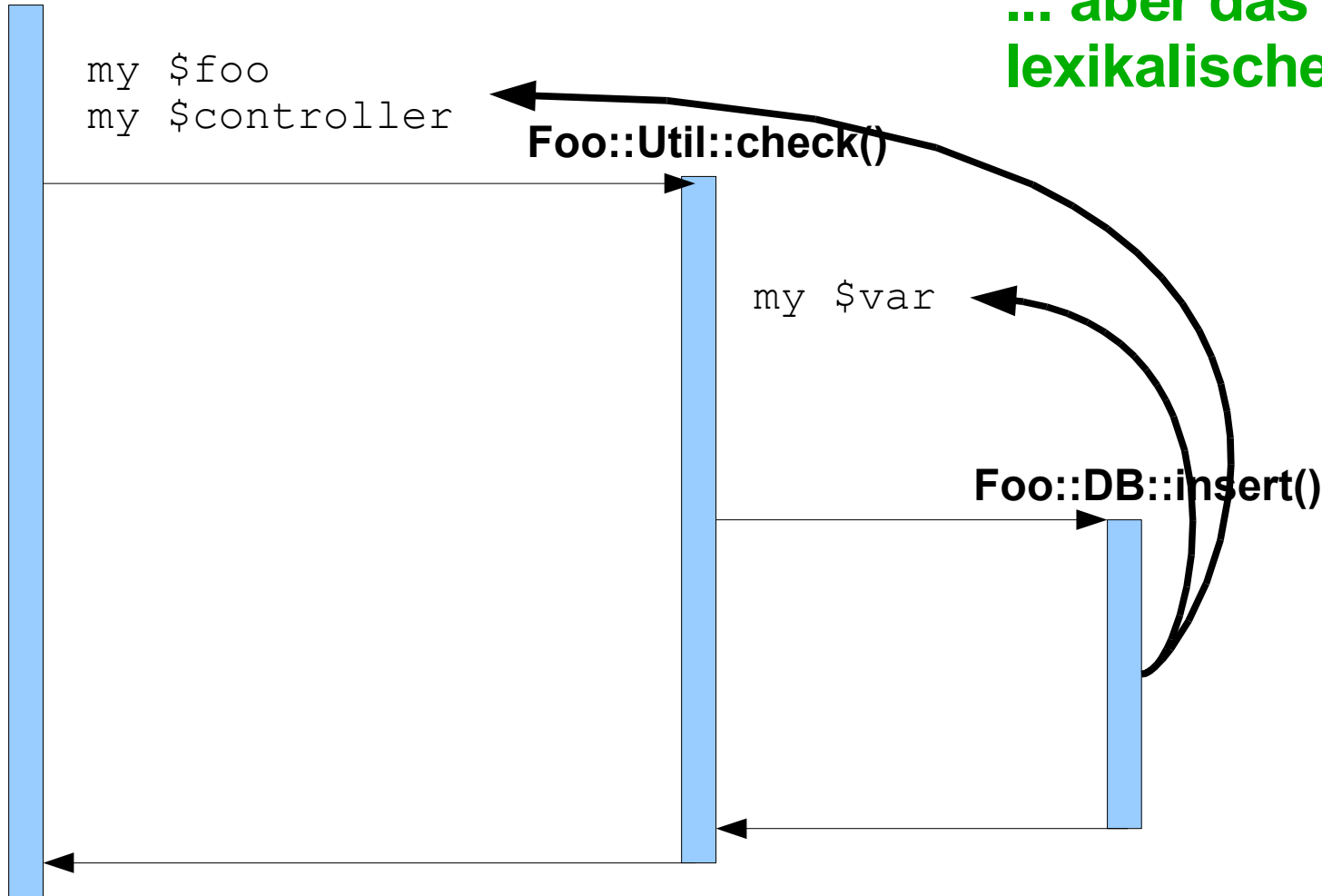
my \$foo
my \$controller

Foo::Util::check()

my \$var

Foo::DB::insert()

... aber das sind
lexikalische Variable



PadWalker

peek_my(\$level)

holt die „my“-Variablen vom Level **\$level**

peek_our(\$level)

holt die „our“-Variablen vom Level **\$level**

```
use PadWalker;
use Data::Dumper;

level1();

sub level1 {
    my $c = { test => 1, hallo => 2, };
    level2();
}

sub level2 {
    my $c = [1,2];

    my $h = PadWalker::peek_my(0);
    print "level2: ", Dumper $h->{'$c'};

    my $t = PadWalker::peek_my(1);
    print "level1: ", Dumper $t->{'$c'};
}
```

```
use PadWalker;
use Data::Dumper;

level1();

sub level1 {
    my $c = { test => 1, hallo => 2, };
    level2();
}

sub level2 {
    my $c = [1,2];

    my $h = PadWalker::peek_my(0);
    print "level2: ", Dumper $h->{'$c'};

    my $t = PadWalker::peek_my(1);
    print "level1: ", Dumper $t->{'$c'};
}
```

```
level2: $VAR1 = \  
          1,  
          2  
        ];  
level1: $VAR1 = \  
          'test' => 1,  
          'hallo' => 2  
        ];
```

Datenbank-Logging

- Häufige Fehlerursache: Datenbankabfragen
 - Nicht existierende Spalten
 - Typos
 - Nicht erlaubte Werte

Datenbank-Logging

- DBIx::Log4perl für plain-DBI
- DBIx::Class::Log4perl für DBIx::Class
 - Noch nicht auf CPAN

DBIx::Log4perl

```
#!/usr/bin/perl

use Log::Log4perl;
use DBIx::Log4perl;

Log::Log4perl->init( 'dbi.conf' );
my $dbh = DBIx::Log4perl->connect( 'DBI:SQLite:dbix_test' )
    or die "Fehler";

my $sth = $dbh->prepare( "SELECT * FROM tabelle WHERE ID = ?" );
$sth->execute(8);

my $dump = do{ local (@ARGV, $/) = $0; <> };

$sth = $dbh->prepare(
    "INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )" );
$sth->execute( 'tester', $dump );
```

DBIx::Log4perl

```
#!/usr/bin/perl

use Log::Log4perl;
use DBIx::Log4perl;

Log::Log4perl->init( 'dbi.conf' );
my $dbh = DBIx::Log4perl->connect( 'DBI:SQLite:dbix_test' )
    or die "Fehler";

my $sth = $dbh->prepare( "SELECT * FROM tabelle WHERE ID = ?" );
$sth->execute(8);

my $dump = do{ local (@ARGV, $/) = $0; <> };

$sth = $dbh->prepare(
    "INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )" );
$sth->execute( 'tester', $dump );
```

DBIx::Log4perl

```
log4perl.logger = DEBUG, T
log4perl.appender.T = \
    Log::Log4perl::Appender::File
log4perl.appender.T.filename = \
    test.log
log4perl.appender.T.layout = \
    PatternLayout
log4perl.appender.T.layout.ConversionPattern = \
    %p - %m %n

use DBIx::Log4perl;
Log::Log4perl->init( 'dbi.conf' );
my $dbh = DBIx::Log4perl->connect( 'DBI:SQLite:dbix_test' );

my $sth = $dbh->prepare( "SELECT * FROM tabelle WHERE ID = ?" );
my $dump = do{ local (@ARGV, $/) = $0; <> };

$sth = $dbh->prepare(
    "INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )" );
$sth->execute( 'tester', $dump );
```

DBIx::Log4perl

```
DEBUG - connect(0): DBI:SQLite:dbix_test,  
INFO - DBI: 1.57, DBIx::Log4perl: 0.13, Driver: SQLite(1.13)  
DEBUG - prepare(0.0): 'SELECT * FROM tabelle WHERE ID = ?'  
DEBUG - execute(0.0) (SELECT * FROM tabelle WHERE ID = ?): 8  
DEBUG - prepare(0.1): 'INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )'  
DEBUG - $execute(0.1) (INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )) =  
    ['tester', '#!/usr/bin/perl ...'];
```

DBIx::Log4perl

```
DEBUG - connect(0): DBI:SQLite:dbix_test,  
INFO - DBI: 1.57, DBIx::Log4perl: 0.13, Driver: SQLite(1.13)  
DEBUG - prepare(0.0): 'SELECT * FROM tabelle WHERE ID = ?'  
DEBUG - execute(0.0) (SELECT * FROM tabelle WHERE ID = ?): 8  
DEBUG - prepare(0.1): 'INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )'  
DEBUG - $execute(0.1) (INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )) =  
    ['tester', '#!/usr/bin/perl ...'];
```

DBIx::Log4perl

```
DEBUG - connect(0): DBI:SQLite:dbix_test,  
INFO - DBI: 1.57, DBIx::Log4perl: 0.13, Driver: SQLite(1.13)  
DEBUG - prepare(0.0): 'SELECT * FROM tabelle WHERE ID = ?'  
DEBUG - execute(0.0) (SELECT * FROM tabelle WHERE ID = ?): 8  
DEBUG - prepare(0.1): 'INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )'  
DEBUG - $execute(0.1) (INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )) =  
    ['tester', '#!/usr/bin/perl ...'];
```

DBIx::Log4perl

```
DEBUG - connect(0): DBI:SQLite:dbix_test,  
INFO - DBI: 1.57, DBIx::Log4perl: 0.13, Driver: SQLite(1.13)  
DEBUG - prepare(0.0): 'SELECT * FROM tabelle WHERE ID = ?'  
DEBUG - execute(0.0) (SELECT * FROM tabelle WHERE ID = ?): 8  
DEBUG - prepare(0.1): 'INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )'  
DEBUG - $execute(0.1) (INSERT INTO tabelle( username, dump ) VALUES ( ?, ? )) =  
    ['tester', '#!/usr/bin/perl ...'];
```

DBIx::Log4perl

[2008/11/12 12:27:46] [DBIx::Log4perl::db] FATAL -

```
=====
DBD::SQLite::db prepare failed: 2 values for 3 columns(1) at dbdimp.c line 271
lasth Statement (DBIx::Log4perl::db=HASH(0x1e3ec58)):
  INSERT INTO tabelle( username, dump, test ) VALUES ( ?, ? )
DB: dbix_test, Username:
handle type: db
SQL: 'Possible SQL: '
db Kids=1, ActiveKids=1
DB errstr: 2 values for 3 columns(1) at dbdimp.c line 271
1 sub statements:
stmt(DBIx::Log4perl::st=HASH(0x1e3ef94)):
DBI error trap at C:/usr/site/lib/DBIx/Log4perl/db.pm line 32
  DBIx::Log4perl::db::prepare('DBIx::Log4perl::db=HASH(0x1e3eca0)',
  'INSERT INTO tabelle( username, dump, test ) VALUES ( ?, ? )') called at
  dbix_log4perl.pl line 14
=====
```

B::Concise

- Nicht direkt zum Debuggen, aber ganz nützlich um „komisches“ Verhalten zu erklären...

B::Concise

```
my $link = 'htt://foo-magazin.de';  
if (! $link =~ m!^https?:://!i ){  
    print 'no http uri';  
}  
else {  
    print 'http uri';  
}
```

http uri

or **no http uri**

```
C:>perl uri.pl  
http uri
```

B::Concise

- Gerade „Einsteiger“ haben mit den Prioritäten ihre Problem...

B::Concise

- ... aber da kann ein Blick in die Ausgabe von B::Concise helfen...

B::Concise

```
C:\>perl -MO=Concise,-exec uri.pl
1 <0> enter
2 <;> nextstate(main 1 uri.pl:3) v
3 <$> const[PV "htt://foo-magazin.de"] s
4 <0> padsv[$link:1,5] sRM*/LVINTRO
5 <2> sassign vKS/2
6 <;> nextstate(main 5 uri.pl:4) v
7 <0> padsv[$link:1,5] s
8 <1> not sK/1
9 </> match(/"^\https?:\/\/"/) sKS/RTIME
a <|> cond_expr(other->b) vK/1
b   <0> pushmark s
c   <$> const[PV "no http uri"] s
d   <@> print vK
      goto e
f <0> enter v
g <;> nextstate(main 3 uri.pl:8) v
h <0> pushmark s
i <$> const[PV "http uri"] s
j <@> print vK
k <@> leave vKP
e <@> leave[1 ref] vKP/REFC
uri.pl syntax OK
```

B::Concise

```
C:\>perl -MO=Concise,-exec uri.pl
...
6 <;> nextstate(main 5 uri.pl:4) v
7 <0> padsv[$link:1,5] s
8 <1> not sK/1
9 </> match(/"^\https?:\/\/"/) sKS/RTIME
a <|> cond_expr(other->b) vK/1
b     <0> pushmark s
c     <$> const[PV "no http uri"] s
d     <@> print vK
        goto e
f <0> enter v
...
```

B::Concise

```
C:\>perl -MO=Concise,-exec uri.pl
...
6 <;> nextstate(main 5 uri.pl:4) v
7 <0> padsv[$link:1,5] s
8 <1> not sK/1
9 </> match(/^https?:\/\//) sKS/RTIME
a <|> cond_expr(other->b) vK/1
b     <0> pushmark s
c     <$> const[PV "no http uri"] s
d     <@> print vK
        goto e
f <0> enter v
...
```

B::Concise

```
7 <0> padsv[$link:1,5] s
8 <1> not sK/1
9 </> match("/^https?://"/) sKS/RTIME
```

... im Vergleich dazu ...

B::Concise

```
my $link = 'htt://foo-magazin.de';
if (not $link =~ m!^https?://!i ) {
    print 'no http uri';
}
else {
    print 'http uri';
}
```

B::Concise

```
7 <0> padsv[$link:1,5] s
8 </> match("/^https?:://" /) sKS/RTIME
9 <1> not sK/1
```

((time - \$^T)

> 25 * 60) ?

“Danke“ :

continue;

DBIx::Class::Log4perl

- Loggt Statements und Parameter mit
- Nutzt den „debugger“

DBIx::Class::Log4perl

```
package My::Schema;  
  
use DBIx::Class::Log4perl;  
use base qw/DBIx::Class::Schema/;  
  
__PACKAGE__->logger_conf( './test2.conf' );  
# load classes  
  
1;
```

DBIx::Class::Log4perl

```
package My::Schema;
```

```
use DBIx::Class::Log4perl;
```

```
use base qw/DBIx::Class::Schema/;
```

```
__PACKAGE__->logger_conf( './test2.conf' );
```

```
# load classes
```

```
1;
```

```
log4perl.logger = DEBUG, Stdout
```

```
log4perl.appender.Stdout = \
```

```
    Log::Log4perl::Appender::File
```

```
log4perl.appender.Stdout.filename = test.log
```

```
log4perl.appender.Stdout.layout = PatternLayout
```

```
log4perl.appender.Stdout.layout.ConversionPattern
```

```
    = %m%n
```

DBIx::Class::Log4perl

```
use My::Schema
```

```
my $schema = My::Schema->connect(  
    'DBI:SQLite:db',  
);
```

```
$schema->logging(1);
```

```
my ($user) = $schema->resultset( 'Test' )  
    ->search({  
    id => 1,  
});
```

DBIx::Class::Log4perl

```
use My::Schema
```

```
my $schema = My::Schema->connect(  
    'DBI:SQLite:db',  
);
```

```
$schema->logging(1);
```

```
my ($user) = $schema->resultset( 'Test' )  
    ->search({  
    id => 1,  
});
```

DBIx::Class::Log4perl

```
C:>perl logging.pl  
[DEBUG] [DBIx::Class::Log4perl] SELECT me.testid,  
        me.name FROM T me  
        WHERE ( testid = ? )  
[DEBUG] [DBIx::Class::Log4perl] '1'
```