Herbert Breunung

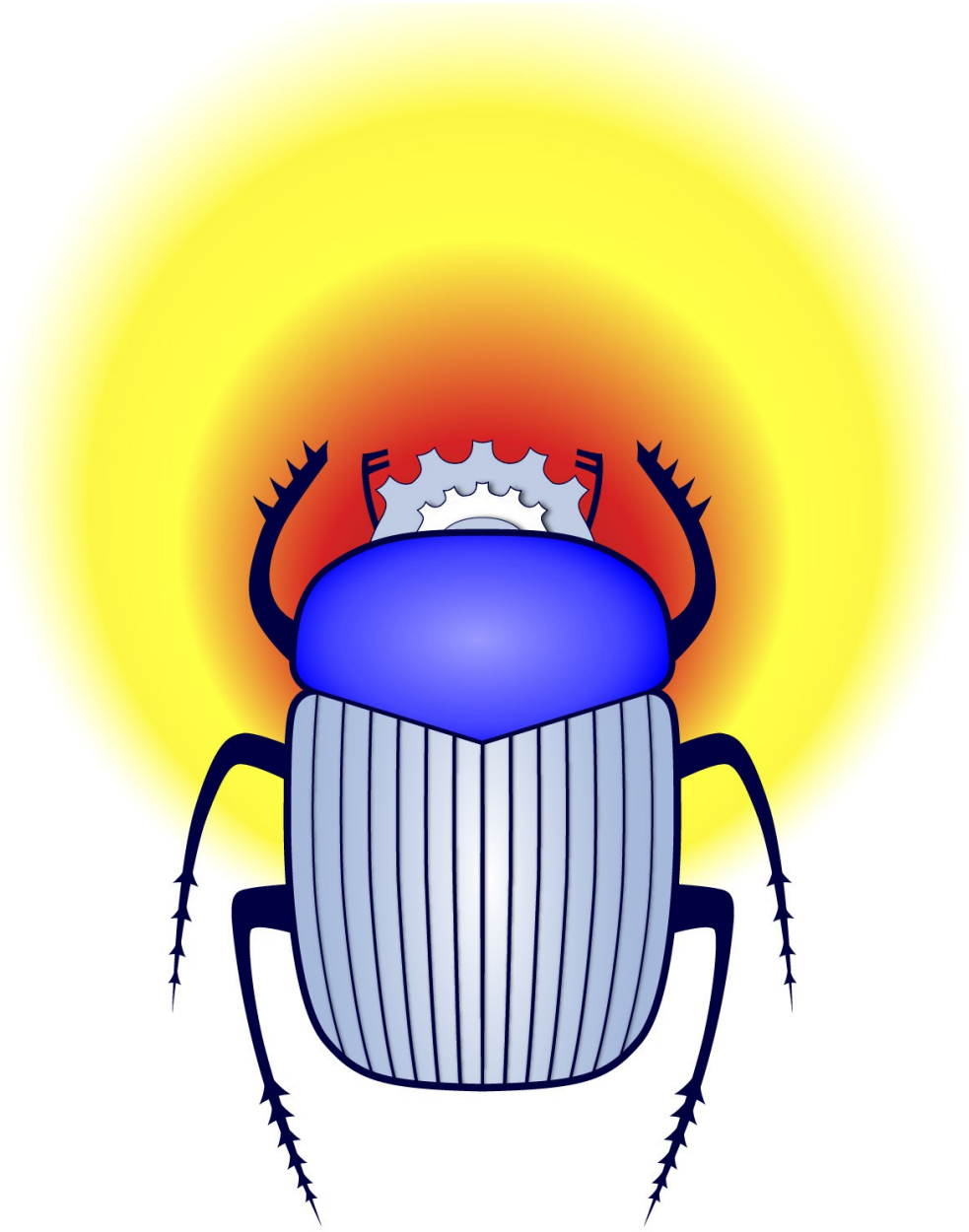
Aus dem Leben eines Perl 6 Modulautors

Herbert Breunung

Autor, Moderator

Votrag. Program.

Kein Kommentar



P6::Math::Matrix

Aus dem Leben eines Perl 6 Modulautors

P6::Math::Matrix

Perl 6 Modulautor

1.P6 Distro

Perl 6 Modulautor

1. P6 Distro
2. PM6 PBP

Perl 6 Modulautor

1. P6 Distro
2. PM6 PBP
3. P6 Anders

1.P6 Distro

P6 Module

7 Core

1077 Σ

Alles erklärt

modules.perl6.org

p6c.org

create a module

modules.perl6.org

Hosting

Github → CPAN

171 results

modules.perl6.org/search/?q=from:cpan

Distribution Development Tool

Schnellstart per ddt

orange == optional

/t

/lib

/bin

/hooks

Dateien

/CHANGELOG.md
/LICENSE
/META6.json
/README.md
/.travis.yml
/.appveyor.yml

Dateien

/lib/Math/Matrix.pm6
/lib/Math/Matrix.pod
/LICENSE
/META6.json
/README.md

LICENSE

The Artistic License 2

.travis.yml

```
language: perl6
perl6:
    - latest
install:
    - rakudobrew build zef
    - zef install --deps-only .
    - zef install Test::META
```

.travis.yml

```
script:
  - PERL6_TEST_META=1
    PERL6LIB=$PWD/lib prove
    -e perl6 -r t/
sudo: false
```

.appveyor.yml

```
os: Visual Studio 2015

platform: x64

install:
  - '"C:\Program Files\Microsoft SDKs\Windows\v7.1\Bin\SetEnv.cmd" /x64'
  - choco install strawberryperl
  - SET PATH=C:\strawberry\c\bin;C:\strawberry\perl\site\bin;C:\strawberry\perl\bin;
%PATH%
  - git clone https://github.com/tadzik/rakudobrew %USERPROFILE%\rakudobrew
  - SET PATH=%USERPROFILE%\rakudobrew\bin;%PATH%
  - rakudobrew build moar 2017.06
  - rakudobrew build zef
  - cd %APPVEYOR_BUILD_FOLDER%
  - zef --verbose --deps-only install .

build: off

test_script:
  - prove -v -e "perl6 -Ilib" t/

shallow_clone: true
```

README.md

. . . . .

CHANGELOG.md

. . . . .

META6.json

http://design.perl6.org/S22.html#META6.json

META6.json

auth authors description depends emulates license name perl production provides resources source-url supersedes superseded-by support tags test-depends version

META6.json

```
{
    "perl"        : "6.c",
    "name"        : "Math::Matrix",
    "license"     : "Artistic-2.0",
    "version"     : "0.2.0",
    "description" : "...",
}
```

META6.json

depends : ["AttrX::Lazy"],

"source-url" :
"git://github.com/pierre-vigier/Perl6-Math-Matrix.git"

META6.json

```
"authors" : [
    "github:..",
    "github:lichtkind"
],
```

META6.json

```
“provides” : {
   "Math::Matrix"      :
        "lib/Math/Matrix.pm6",
```

META6.json

“tags” : ["Math",]

Tests: /t/*.t

```
use Test;
use Math::Matrix;

plan 1;
ok $m.cell(0,0) ~~ 4, "...";
```

Testgruppen

```
use Test;
plan 1;
subtest {
    plan 12;

    ...
}, "Thema";
```

Test::Exception

```
use Test;
use Math::Matrix;

plan 1;
dies-ok { ... } , "Fehler...";
```

Test.pm

plan done-testing bail-out todo skip skip-rest diag subtest pass flunk ok nok cmp-ok is is-deeply isnt is-approx like unlike use-ok isa-ok does-ok can-ok dies-ok lives-ok eval-dies-ok eval-lives-ok throws-like

Andere /t Module

plan done-testing bail-out todo skip skip-rest diag subtest pass flunk ok nok cmp-ok is is-deeply isnt is-approx like unlike use-ok isa-ok does-ok can-ok dies-ok lives-ok eval-dies-ok eval-lives-ok throws-like

Test::META

use it

Test Module

Test::META
Test::When
Test::Output
Test::IO::Socket
Test::Util::ServerPort

Test Module

Test::Mock
Test::Lab
TestML

use zef

zef install Math::Matrix --force

2.PM6 PBP

Wie Zuhaus

use name;

require name;

Fast Wie Zuhaus

use name;

need name;

require name;

Module

use v6.c;

unit module name;

class name::space{...}

Was ist unit?

unit class Math::Matrix..;
unit role ...;
unit module ...;

Version und Quelle

```
unit class Math::Matrix
:ver<0.1.8>
:auth<github:pierre-vigier>;
```

eignes Modul

```
use AttrX::Lazy;
```

Eigne Typ

```
subset PosInt of Int where * > 0;
```

kann weniger

```
subset PosInt of Int where * > 0;

$rows where * > 0,
```

Am Besten

```
subset PosInt of Int where * > 0;

$rows where * > 0,

PosInt $rows,
```

Besser als

```
@rows where .all ~~ Numeric,
```

Profityp

```
subset NumList of List
        where .all ~~ Numeric;

fail "..."
    unless @rows ~~ NumList;
```

2D

@rows **where** .all ~~ Numeric,

all(@m[*;*]) ~~ Numeric;

Notlösung

```
submethod check_index
(Int $row, Int $col) {,
```

Unterscheide

```
method name          # normal
method !name         # privat
submethod name       # geizig
sub name              # no OO
```

nichts besonderes

```
method new (@rows) {
    self.bless(...);
}
```

schon besonders

```
submethod BUILD (...) {
    @!rows = ;
}
```

Neutrales Element

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Neutrales Element

```
method new-identity
  (Math::Matrix:U: Int $I -->
    self.bless(...);
}
```

Typobjekt

```
method new-identity
 (Math::Matrix:U: Int $I -->
   self.bless(...);
}
```

Typobjekt

```
method new-identity
 (Math::Matrix:U: Int $I -->
   self.bless(...);
}
Math::Matrix.new-identity(3);
```

Typobjekt

```
method new-identity
 (Math::Matrix:D: Int $I -->
   self.bless(...);
}
$matrix.new-identity(3);
```

Null

0 0 0

0 0 0

0 0 0

Null

```
method new-zero
  (Math::Matrix:U:
   PInt $rows, PInt $rows -->
    self.bless(...);
}
```

Constructor

default new(....)

schnell, einfach, no prop.

besonders new-....()

eigen sigs., props!

custom BUILD

wird von beidem gerufen

Norm

default p q

p, q optional

besondere namen

in sig abgefragt

MMD

jede norm - eigne multi

p-q Norm

```
multi method norm
( PInt :$p = 2, PInt :$q = 1
--> Numeric) { ...
```

Norm

```
multi method norm
(Str $which where
* eq 'row-sum' --> Numeric) {
    max map {
        [+] map { abs $_ }, @$_
    }, @!rows;
}
```

cloning

```
method clone {
  self.bless(rows =>@!rows)
}
```

no deep clones

```
method clone {
  self.bless(rows =>@!rows)
}
```

marshal

```
multi method perl
 (Math::Matrix:D: --> Str) {
    self.WHAT.perl  ~
  ".new(" ~ @!rows.perl ~ ")";
}
```

multi is important

```
multi method perl
 (Math::Matrix:D: --> Str) {
    self.WHAT.perl  ~
  ".new(" ~ @!rows.perl ~ ")";
}
```

~ $matrix

```
method Str
    (Math::Matrix:D: --> Str) {
        ...
}
```

? $matrix

```
method Bool
 (Math::Matrix:D: --> Bool) {
    ! self.is-zero;
}
```

is Boolean

```
method is-square
  (Math::Matrix:D: --> Bool) {
    ...
}
```

+ $matrix

```
method Numeric
 (Math::Matrix:D: --> Bool) {
    self.elems;
}
```

say $matrix

```
method gist
  (Math::Matrix:D: --> Str) {
      ...
}
```

```
say $matrix.full

method full
  (Math::Matrix:D: --> Str) {
     @!rows.gist;
}
```

Smartmatch: $a ~~ $b

```
method ACCEPTS
(Math::Matrix:D:
 Math::Matrix $b --> Bool) {
    self.equal( $b );
}
```

$a ~~ Math::Matrix

```
multi method ACCEPTS
(Math::Matrix:D:
 Math::Matrix $b --> Bool) {
    self.equal( $b );
}
```

$matrix.map({...})

```
method map
(Math::Matrix:D: &coderef
            --> Math::Matrix:D) {
    Math::Matrix.new(
        [ @!rows.map:
  {[ $_.map( &coderef ) ] } ] );
}
```

Liste → Liste

```
$matrix.list-rows.map(...
$matrix.list-rows.flat.map(...

$matrix.map(...
```

Matrix → Matrix

$matrix.reduce({...})

method reduce **...**

$m.reduce-rows(&[+])

method reduce ...

method reduce-rows ...

method reduce-columns ...

$matrix.grep(&[+])

method grep … ?

Operators

sub name **is export** { … }

Op: + - *

```
multi sub infix:<+>
  (Math::Matrix:D $a,
   Math::Matrix:D $b -->
   Math::Matrix:D) is export {
    $a.add($b);
}
```

Norm: || $matrix ||

```
multi sub circumfix:<|| ||>
  (Math::Matrix $a --> Numeric)
is equiv(&prefix:<!>) is export {
    $a.norm();
}
```

3.P6 Anders

Vergaß unit

```
unit class Math::Matrix..;
unit role ...;
unit module ...;
```

Kindersicherung

```
use lib '.';
use Math::Matrix;
```

Einfacher Port

eval "...";

nagut …

EVAL "…";

Kindersicherung

```
use MONKEY;
use MONKEY-SEE-NO-EVAL;

EVALFILE $file;
EVAL "...";
```

Namensräume

```
class Math::Matrix.. {}
role … {}
module … {}
```

Vergaß unit

```
unit class Math::Matrix..;
unit role ...;
unit module ...;
```

Semikolon !

```
sub g { 9.81 }
sub e { 2.718 } # e is da

sub g { 9 }; sub d { 2 }
```

Routinen

g() + d() : 11

g() - d() : 7

ohne & und ()

g() + d : 11

g() - d : 7

Mehrdeutig !

g + d : Err

g - d : Err

Eindeutig !

g + d() : 11

g - d() : 7

Leerzeichen !

g+d : 11

g-d : Err

besonderes: -

$!row-count - 1
$!row-count-1
$!row-count-d

Konversionen

```
method column ...
@!rows.keys.map:{
    @!rows[$_;$column]
};
```

Seq vs List

```
method column
(Math::Matrix:D:
 Int:D $column --> List) {
    @!rows.keys.map:{
        @!rows[$_;$column]
    };
```

Seq vs List

```
method column
(Math::Matrix:D:
 Int:D $column --> List) {
    (@!rows.keys.map:{
        @!rows[$_;$column]
    }).list;
```

Danke

FIN

lichtkind.de/vortrag