Perl Mongers im Ruhrgebiet

Ruhr.pm

Class::DBI::AbstractSearch

Autor: Simon A. Wilper

EMail: simon AT ruhr.pm.org

Datum: 13. August 2007

http://ruhr.pm.org/

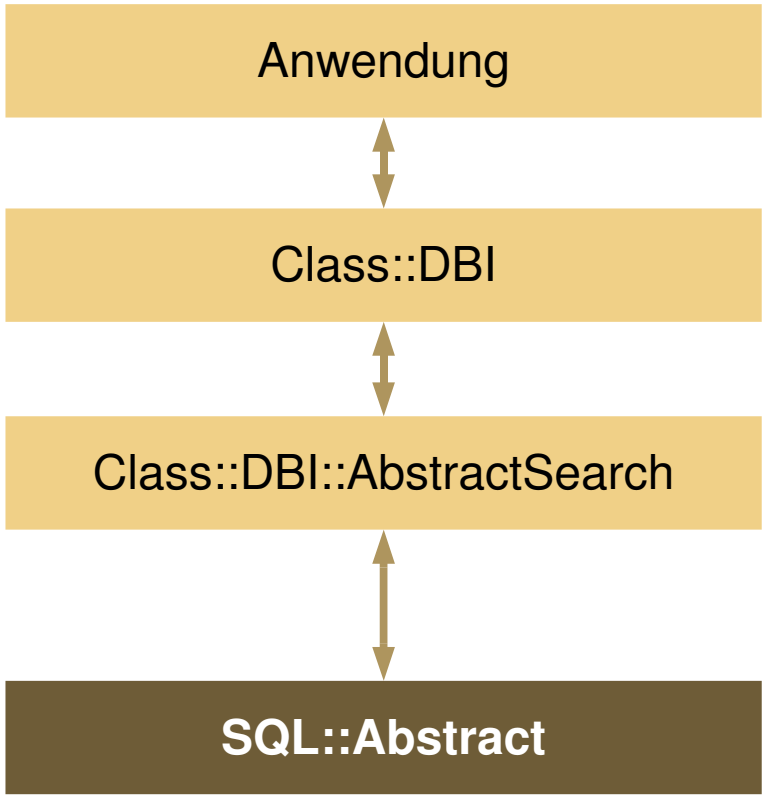
Ruhr.pm

Was ist Class::DBI::AbstractSearch?

- Abstraktionsschicht fuer SQL-Befehle
- Plugin fuer Class::DBI
- Benutzt:
  - SQL::Abstract ("Konverter" von Perl-Datenstrukturen nach SQL.)
  - SQL::Limit (unterstuetzt verschiedene LIMIT-Dialekte)

Ruhr.pm

Auf bau



Beispiel (Wdh. aus letztem Vortrag)

```
#!/usr/bin/perl

use warnings;
use strict;

# ---
package Addresses::DBI;
use base 'Class::DBI';

Addresses::DBI->connection(
    'dbi:Pg:dbname=address', # DBName
    '',                      # User
    '',                      # Password
    { AutoCommit => 1 }      # Options
);
```

```
# ---
package Addresses::Names;
use base 'Addresses::DBI';
use Class::DBI::AbstractSearch;

Addresses::Names->table( 'names' );

Addresses::Names->columns(
        All => qw(id first last email
                department task)
);

# ---
sub PrintRow {
    my( $Row ) = @_;

    printf( "% 5d %-20s %-20s".
                " %-30s % 5d %s\n",
        $Row->id,   $Row->first,
        $Row->last, $Row->email,
        $Row->department,
        $Row->task
    );
}
```

Ruhr.pm

Auf r uf

- Class::DBI::AbstractSearch Methode:
  `search_where( \%where, \%attrs )`
  - `\%where`: Struktur, die die WHERE-Clause abbildet
  - `\%attrs`: Struktur fuer weitere Optionen:
    - `order_by`
    - `limit_dialect`
    - `limit`
    - `offset`

Ruhr.pm

Tabellenauszug "names"

id	first	last	email	department	task
11	John	Dow	dow@nowhere.org	140	design
12	Mike	Dowell	mdowell@company.org	140	code
13	John	Smith	jsmith@company.org	150	sound
14	Henry	Mcenroe	hmcenroe@company.org	170	code
15	John	Wayne	wayne@whoknows.net	150	qa

OR-Abfrage

- Oder-Verknuepfungen abgebildet durch anonyme Listen

```
sub SearchWhere {
      my $Iterator = Addresses::Names->search_where(
            { department => [ 140, 150 ] }
      );

      while ( my $Row = $Iterator->next ) {
            PrintRow( $Row );
      }
}
```

Ruhr.pm

Resultat

```
where department = 140 or department = 150
```

id	first	last	email	department	task
11	John	Dow	dow@nowhere.org	140	design
12	Mike	Dowell	mdowell@company.org	140	code
13	John	Smith	jsmith@company.org	150	sound
~~14~~	~~Henry~~	~~Mcenroe~~	~~hmcenroe@company.org~~	~~170~~	~~code~~
15	John	Wayne	wayne@whoknows.net	150	qa

AND-Abfrage

- Und-Verknuepfungen abgebildet durch anonyme Hashes

```
sub SearchWhere {
      my $Iterator = Addresses::Names->search_where(
            { department => [ 140, 150 ], task => 'design' }
      );

      while ( my $Row = $Iterator->next ) {
            PrintRow( $Row );
      }
}
```

Ruhr.pm

Resultat

```
where ( department = 140 or department = 150 )
and task = 'design'
```

id	first	last	email	department	task
11	John	Dow	dow@nowhere.org	140	design
~~12~~	~~Mike~~	~~Dowell~~	~~mdowell@company.org~~	~~140~~	~~code~~
~~13~~	~~John~~	~~Smith~~	~~jsmith@company.org~~	~~150~~	~~sound~~
~~14~~	~~Henry~~	~~Mcenroe~~	~~hmcenroe@company.org~~	~~170~~	~~code~~
~~15~~	~~John~~	~~Wayne~~	~~wayne@whoknows.net~~	~~150~~	~~qa~~

Ruhr.pm

Bereiche abfragen

- Operator wird zusammen mit dem Wert in einem Hash uebergeben

```
my $Iterator = Addresses::Names->search_where(
      { department => { '>' => 140, '<' => 170 } }
);
```

Ruhr.pm

Weitere typische SQL-Konstrukte

- LIKE — last => { -like => 'Dow%' }
- BETWEEN — id => { -between => [12, 15] }
- NOT BETWEEN — id => { -not_between => [12, 15] }

Nativen SQL-Befehl ausgeben

- SQL zur Vorschau ausgeben
- Wiederverwendbarkeit
- Rueckgabe $sql und @bind

```
my $sqlAbs = new SQL::Abstract;

my ( $sql, @bind ) = $sqlAbs->where(
      $where, $attrs->{'order_by'}
);

print "Native SQL: $sql\nBind Values: ",
join( ',', @bind ), "\n";
```

Fragen?