[![Build Status](https://travis-ci.org/gfx/p5-Hash-FieldHash.svg?branch=master)](https://travis-ci.org/gfx/p5-Hash-FieldHash)

NAME

Hash::FieldHash - Lightweight field hash for inside-out objects

VERSION

This document describes Hash::FieldHash version 0.15.

SYNOPSIS

```
use Hash::FieldHash qw(:all);

fieldhash my %foo;

fieldhashes \my(%bar, %baz);

{
        my $o = Something->new();

        $foo{$o} = 42;

        print $foo{$o}; # => 42
}
# when $o is released, $foo{$o} is also deleted,
# so %foo is empty in here.

# in a class
{
        package Foo;
        use Hash::FieldHash qw(:all);

        fieldhash my %bar, 'bar'; # make an accessor
}

my $obj = bless {}, 'Foo';
$obj->bar(10); # does $bar{$obj} = 10
```

DESCRIPTION

`Hash::FieldHash` provides the field hash mechanism which supports the inside-out technique.

You may know `Hash::Util::FieldHash`. It's a very useful module, but too complex to understand the functionality and only available in 5.10. `H::U::F::Compat` is available for pre-5.10, but it is too slow to use.

This is a better alternative to `H::U::F` with following features:

Simpler interface
: `Hash::FieldHash` provides a few functions: `fieldhash()` and `fieldhashes()`. That's enough.

Higher performance
: `Hash::FieldHash` is faster than `Hash::Util::FieldHash`, because its internals use simpler structures.

Relic support
: Although `Hash::FieldHash` uses a new feature introduced in Perl 5.10, *the uvar magic for hashes* described in "GUTS" in Hash::Util::Fieldhash, it supports Perl 5.8 using the traditional tie-hash layer.

INTERFACE

Exportable functions

`fieldhash(%hash, ?$name, ?$package)`
Creates a field hash. The first argument must be a hash.

Optional *$name* and *$package* indicate the name of the field, which will create rw-accessors, using the same name as *$name*.

Returns nothing.

`fieldhashes(@hash_refs)`
Creates a number of field hashes. All the arguments must be hash references.

Returns nothing.

`from_hash($object, \%fields)`
Fills the named fields associated with *$object* with *%fields*. The keys of *%fields* can be simple or fully qualified.

Returns *$object*.

`to_hash($object, ?-fully_qualify)`
Serializes *$object* into a hash reference.

If the `-fully_qualify` option is supplied , field keys are fully qualified.

For example:

```
package MyClass;
use FieldHash qw(:all);

fieldhash my %foo => 'foo';

sub new{
        my $class = shift;
        my $self  = bless {}, $class;
        return from_hash($self, @_);
}

package MyDerivedClass;
use parent -norequire => 'MyClass';
use FieldHash qw(:all);

fieldhash my %bar => 'bar';

package main;

my $o = MyDerivedClass->new(foo => 10, bar => 20);
my $p = MyDerivedClass->new('MyClass::foo' => 10, 'MyDerivedClass::bar' =

use Data::Dumper;
print Dumper($o->to_hash());
# $VAR1 = { foo => 10, bar => 20 }

print Dumper($o->to_hash(-fully_qualify));
# $VAR1 = { 'MyClass::foo' => 10, 'MyDerived::bar' => 20 }
```

ROBUSTNESS

Thread support

As `Hash::Util::FieldHash` does, `Hash::FieldHash` fully supports threading using the `CLONE` method.

Memory leaks

`Hash::FieldHash` itself does not leak memory, but it may leak memory when you uses hash references as field hash keys because of an issue of perl 5.10.0.

NOTES

The type of field hash keys

`Hash::FieldHash` accepts only references and registered addresses as its keys, whereas `Hash::Util::FieldHash` accepts any type of scalars.

According to "The Generic Object" in Hash::Util::FieldHash, Non-reference keys in `H::U::F` are used for class fields. That is, all the fields defined by `H::U::F` act as both object fields and class fields by default. It seems confusing; if you do not want them to be class fields, you must check the type of *`$self`* explicitly. In addition, these class fields are never inherited. This behavior seems problematic, so `Hash::FieldHash` restricts the type of keys.

The ID of field hash keys

While `Hash::Util::FieldHash` uses `refaddr` as the IDs of field hash keys, `Hash::FieldHash` allocates arbitrary integers as the IDs.

What accessors return

The accessors `fieldhash()` creates are **chainable** accessors. That is, it returns the *`$object`* (i.e. `$self`) with a parameter, where as it returns the *`$value`* without it.

For example:

```
my $o = YourClass->new();
$o->foo(42);           # returns $o itself
my $value = $o->foo(); # retuns 42
```

DEPENDENCIES

Perl 5.8.5 or later, and a C compiler.

BUGS

No bugs have been reported.

Please report any bugs or feature requests to the author.

SEE ALSO

Hash::Util::FieldHash.

Hash::Util::FieldHash::Compat.

"Magic Virtual Tables" in perlguts.

Class::Std describes the inside-out technique.

AUTHOR

Fuji, Goro (gfx) <gfuji(at)cpan.org>.

LICENSE AND COPYRIGHT