



# Introduction to Class::DBI

Lars Thegler

[lars@thegler.dk](mailto:lars@thegler.dk)



# What is it?

## Database abstraction

- Portable between vendors
- Implement cross-platform functionality

## Object persistence

- Perl objects remain alive across invocations

## Object-to-relational mapping

## Built on DBI



# How does it look?

```
use Pers;  
  
my $empl = Pers::Empl->retrieve(7369); # primary key  
  
print $empl->ename; # prints "Smith"  
print $empl->job; # prints "Clerk"  
  
$empl->job('Manager'); # promote him  
  
$empl->update; # write to DB
```



# Write base class

Base class contains

- Database connection
- Application-wide stuff

```
package Pers::Base;  
use base 'Class::DBI';  
__PACKAGE__->set_db('Main', $dsn, $user, $pw);
```



# Write object classes

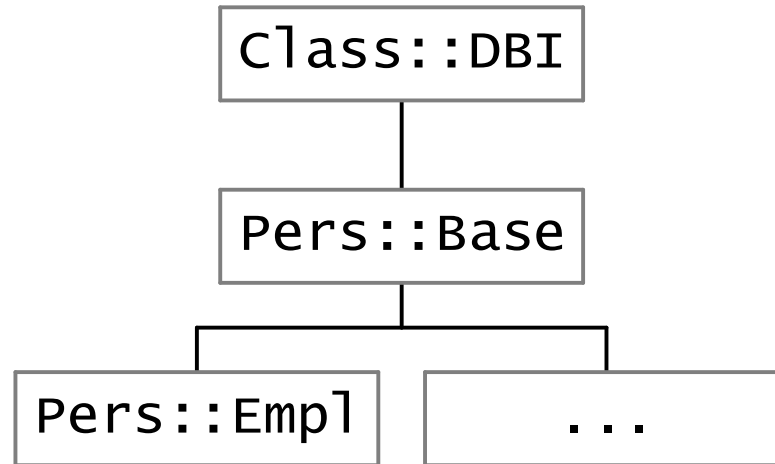
Object class contains

- Table name declaration
- Column name declarations
- Object-specific stuff

```
package Pers::Emp1;  
use base 'Pers::Base';  
__PACKAGE__->table('emp');  
__PACKAGE__->columns(All => ('empno', 'ename', 'job'));
```



# Class Hierarchy





# What have I got now?

## Constructor/Destructor

```
$obj = Pers::Emp1->create(...);  
$obj->delete;
```

## Accessors for all columns

```
$val = $obj->column;  
$obj->column(...);
```

## Fetching

```
$obj = Pers::Emp1->retrieve(...);  
@objs = Pers::Emp1->search(...);
```



# Create

Make a new object

```
my $attrs = {empno => 7654,  
             ename => 'Jones',  
             job   => 'Analyst'};
```

```
my $emp1 = Pers::Emp1->create($attrs);
```

```
$emp1->update;
```



# Delete

Deleting the obsolete

```
$emp1->delete;
```



# Accessors

## Set one field

```
$emp1->job('Manager');
```

## Set multiple fields

```
$emp1->set(job => 'Manager',  
          dept => '30');
```

## Get multiple fields

```
($name, $job) = $emp1->get('ename', 'job');
```



# Fetching

## Get one

```
my $empl = Pers::Empl->retrieve(7654); # primary key
```

## Get all

```
my @staff = Pers::Empl->retrieve_all;
```

## Find matching objects

```
my @clerks = Pers::Empl->search(job => 'clerk');
```

## Find with wildcard

```
my @people = Pers::Empl->search_like(name => 'A%');
```



# Iterators

Don't fetch everything at once

```
my $iter = Pers::Emp1->retrieve_all;
```

```
while (my $emp = $iter->next) {  
    print $emp->ename, "\n";  
}
```

Uses 'lazy load'



# Relations

## 1-to-many relationship

```
Pers::Dept->has_many(emps => 'Pers::Emp1');
```

```
my @emp1s = $dept->emps;
```

## Many-to-1 relationship

```
Pers::Emp1->has_a(dept => 'Pers::Dept');
```

```
my $dept = $emp1->dept;
```



# Inflation

Convert field value to object

```
Pers::Emp1->has_a(hiredate => Date::Simple);
```

```
print $emp1->hiredate->month, "\n";
```

Also converts back ('deflate')



# Triggers

Full set of trigger points:

```
{before,after}_create  
{before,after}_set_$column  
{before,after}_update  
{before,after}_delete  
select
```

```
__PACKAGE__->add_trigger(before_update => sub {...});
```

This is how constraints are built



# Extending

Add your own accessors

```
__PACKAGE__->add_constructor(meth => $where_clause);
```

```
my @arr = Class->meth;
```

Inline your SQL queries

```
my @arr = Class->retrieve_from_sql($where_clause);
```

Class::DBI inherits from Ima::DBI

```
__PACKAGE__->set_sql('update', $update_clause);
```



# Transactions

Either:

```
__PACKAGE__->set_db('Main', $dsn, $user, $pass,  
                    {AutoCommit => 1});
```

Or:

```
$obj->dbi_commit();  
$obj->dbi_rollback();
```



# Databases supported

Class::DBI::MySQL

Class::DBI::Oracle

Class::DBI::PostgreSQL

Class::DBI::SQLite

Class::DBI::Sybase



# Associated modules

- `Class::DBI::View`
  - Virtual views
- `Class::DBI::AbstractSearch`
  - Do more complex searches (`>=`, `!=`)
- `Class::DBI::ToSax`
  - Turn database objects to SAX events
- `Class::DBI::Pager`
  - Break result set into pages, for displaying
- `Class::DBI::AsForm`
  - Produce HTML form elements for database columns
- `Class::DBI::FromCGI`
  - Update database objects using `CGI::Untaint`
- `Class::DBI::Factory`, `Class::DBI::ConceptSearch`,  
`Class::DBI::DDL`, `Class::DBI::PlugIn::*`, ...



# Further reading

The wiki:

<http://www.class-dbi.com/cgi-bin/wiki/index.cgi>

The mailing list:

<http://groups.kasei.com/mail/info/cdbi-talk>



# Questions?

<http://perlworkshop.dk/2004/presentations/>