Persistent object definitions in depth

eZPersistentObject for PHP developers - every key of definition(), the field forms, the condition syntax of fetchObjectList() with operators, ranges and IN lists, sorting, limits and custom fields, store() and sync()...

Source file: `doc/eng-us/help/php-developers/persistent-object-definitions-in-depth.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/persistent-object-definitions-in-depth.md

This page is for PHP developers who store data in tables of their own. `eZPersistentObject` (`kernel/classes/ezpersistentobject.php`) maps a table to a class through a static `definition()` array and builds the SQL for every supported database. The first steps are on Persistent objects and custom tables; this page is the reference behind them.

Contents

1. The definition keys
2. Field definitions
3. Fetching: the parameters
4. The condition syntax
5. Writing
6. Attributes and templates
7. Common mistakes

The definition keys

Key	Meaning
`fields`	the table's columns: column name => field definition
`keys`	the columns that identify a row (the primary key)
`increment_key`	the column filled by the database on insert; set on the object after `store()`
`function_attributes`	extra attribute names mapped to methods, read with `attribute()` and in templates
`set_functions`	attribute names mapped to methods used by `setAttribute()`
`class_name`	the class instantiated for fetched rows
`sort`	default sorting: column => asc or desc
`name`	the table name
`grouping`	default grouping for list fetches

PHP

```php
class acmeReportItem extends eZPersistentObject
{
    public static function definition()
    {
```

```
        return array(
            'fields' => array(
                'id'      => array( 'name' => 'ID', 'datatype' => 'integer', 'default' => 0, 'req
uired' => true ),
                'user_id' => array( 'name' => 'UserID', 'datatype' => 'integer', 'default' => 0,
'required' => true ),
                'status'  => array( 'name' => 'Status', 'datatype' => 'string', 'default' => 'ope
n', 'required' => true ),
                'amount'  => array( 'name' => 'Amount', 'datatype' => 'integer', 'default' => 0,
'required' => true ),
                'created' => array( 'name' => 'Created', 'datatype' => 'integer', 'default' => 0,
 'required' => true ) ),
            'keys' => array( 'id' ),
            'increment_key' => 'id',
            'function_attributes' => array( 'user' => 'user' ),
            'class_name' => 'acmeReportItem',
            'sort' => array( 'created' => 'desc' ),
            'name' => 'acme_report_item' );
    }

    public function user()
    {
        return eZUser::fetch( $this->attribute( 'user_id' ) );
    }
}
```

The kernel's own classes follow the same shape, for example `eZSection::definition()` in `kernel/classes/ezsection.php`.

Field definitions

A field is either the short form, column => member name (`'id' => 'ID'`), or an array:

Key	Meaning
`name`	the member variable name
`datatype`	`integer`, `float`, `string`, used when values are written
`default`	the value of a new object
`required`	whether the field must be set

Fetching: the parameters

PHP

```
eZPersistentObject::fetchObject( $def, $fieldFilters, $conds, $asObject = true, $grouping = null,
 $customFields = null );
eZPersistentObject::fetchObjectList( $def, $fieldFilters = null, $conds = array(), $sorts = null,
 $limit = null,
                                     $asObject = true, $grouping = false, $customFields = null,
                                     $customTables = null, $customConds = null );
eZPersistentObject::count( $def, $conds = null, $field = null );
```

Parameter	Meaning
`$fieldFilters`	`null` for all columns, or a list of column names
`$conds`	the conditions, see below

Parameter	Meaning
`$sorts`	`null` uses the definition's `sort`, `false` none, or column => asc/desc
`$limit`	`array( 'offset' => 0, 'length' => 20 )`
`$asObject`	`true` for objects of `class_name`, `false` for plain rows
`$customFields`, `$customTables`, `$customConds`	extra columns, tables and a raw `WHERE` part for joins; escape anything you put into `$customConds` yourself

`fetchObjectList()` returns an array, or `null` on a database error: guard before `foreach`.

The condition syntax

Condition	SQL
`array( 'id' => 5 )`	`id = 5`
`array( 'amount' => array( '>', 5 ) )`	`amount > 5`
`array( 'created' => array( false, array( $from, $to ) ) )`	`created BETWEEN $from AND $to`
`array( 'id' => array( array( 1, 5, 7 ) ) )`	`id IN ( 1, 5, 7 )`
several keys	joined with `AND`

Values in conditions are escaped for you; that is the main reason to prefer persistent objects over hand-written SQL (SQL with eZDB, safely). Still cast ids (`(int)$id`), so a wrong type never reaches the query.

Writing

PHP

```php
$item = new acmeReportItem( array( 'user_id' => $userID, 'status' => 'open', 'amount' => 2, 'crea
ted' => time() ) );
$item->store();                                  // insert; 'id' is set from increment_key
$item->setAttribute( 'amount', 3 );
$item->sync();                                   // store only when an attribute changed (dirty
data)
$item->remove();                                 // delete this row
eZPersistentObject::removeObject( acmeReportItem::definition(), array( 'user_id' => (int)$userID
) );
```

`store()`, `sync()` and `remove()` are "transaction unsafe" in the kernel's own words: wrap related writes in one transaction (`eZDB::instance()->begin()` and `commit()`).

Attributes and templates

`attribute( $name )` reads a field or calls the function attribute; `hasAttribute()` says whether a name exists. That interface is what templates use, so a persistent object returned by a fetch function is readable as `$item.status` and `$item.user.login` (Write a fetch function).

Common mistakes

Mistake	Effect	Fix
a SQL string in `$conds`	not supported; use the array syntax	`array( 'amount' => array( '>', 5 ) )`
unbounded `fetchObjectList()`	memory and time grow with the table	always pass `$limit`
`foreach` over the result without a check	warning when the fetch failed (`null`)	`if ( is_array( $rows ) )`
undeclared member properties	PHP 8.2 deprecation notices in old classes	declare the members
several writes without a transaction	half-written data after an error	`begin()` and `commit()` around them

Related pages

- eZPersistentObject reference
- Active record pattern
- Persistent objects and tables
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.