eZPersistentObject

The eZPersistentObject reference - definition() keys, fetchObject(), fetchObjectList() and its parameters, count(), store(), remove(), removeObject(), attribute access - with kernel signatures.

Source file: `doc/eng-us/help/extension-development/part-8-reference/ezpersistentobject.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/extension-development/part-8-reference/ezpersistentobject.md

Part 8, reference 5. `eZPersistentObject` (`kernel/classes/ezpersistentobject.php`) maps a table row to an object. The kernel's own classes (content objects, nodes, users, roles) are built on it, and it is the recommended way to keep your own tables. The tutorial is Persistent objects and your own tables and Your own table with a persistent object, by hand. This page is the reference.

definition()

A `public static function definition()` returns an array:

Key	Meaning
`fields`	column name => `array( 'name' => attribute name, 'datatype' => 'integer'|'string'|'float'|'text', 'default' => ..., 'required' => bool )`
`keys`	the primary key columns
`increment_key`	the auto-increment column, if any
`class_name`	the class rows become
`name`	the table name
`sort`	default sort, `array( column => 'asc'|'desc' )`
`function_attributes`	attribute name => method name, for computed attributes readable through `attribute()` (and templates)
`set_functions`	attribute name => method name, called by `setAttribute()`

Fetching

PHP

```php
public static function fetchObject( $def, $field_filters, $conds, $asObject = true, $grouping = null, $custom_fields = null )

public static function fetchObjectList( $def, $field_filters = null, $conds = [], $sorts = null, $limit = null,
                                        $asObject = true, $grouping = false, $custom_fields = null,
                                        $custom_tables = null, $custom_conds = null )

public static function count( $def, $conds = null, $field = null )
```

Parameter	Meaning
`$def`	your `definition()`
`$field_filters`	the columns to fetch; `null` for all
`$conds`	`array( column => value )`; a value may be `array( '<', 10 )` (also >, <=, >=, <>, like), `array( array( 1, 2, 3 ) )` for IN, or `array( false, array( 200, 500 ) )` for BETWEEN
`$sorts`	`array( column => 'asc' )`; `null` uses the definition's sort
`$limit`	`array( 'offset' => 0, 'limit' => 20 )`
`$asObject`	`false` returns plain arrays

PHP

```php
<?php
class myExtItem extends eZPersistentObject
{
    public static function definition()
    {
        return array(
            'fields' => array(
                'id' => array( 'name' => 'ID', 'datatype' => 'integer', 'default' => 0, 'required
' => true ),
                'parent_id' => array( 'name' => 'ParentID', 'datatype' => 'integer', 'default' =>
 0, 'required' => true ),
                'name' => array( 'name' => 'Name', 'datatype' => 'string', 'default' => '', 'requ
ired' => true ),
            ),
            'keys' => array( 'id' ),
            'increment_key' => 'id',
            'class_name' => 'myExtItem',
            'sort' => array( 'name' => 'asc' ),
            'name' => 'myext_item',
        );
    }

    public static function fetch( $id )
    {
        return eZPersistentObject::fetchObject( self::definition(), null, array( 'id' => (int)$id
 ) );
    }

    public static function fetchByParent( $parentID, $offset = 0, $limit = 20 )
    {
        return eZPersistentObject::fetchObjectList( self::definition(), null, array( 'parent_id'
=> (int)$parentID ),
                                                    null, array( 'offset' => $offset, 'limit' =>
$limit ) );
    }

    public static function countByParent( $parentID )
    {
        return eZPersistentObject::count( self::definition(), array( 'parent_id' => (int)$parentI
D ) );
    }
}
```

`fetchObject()` returns `null` when nothing matches. Condition values are escaped and quoted by the class (as strings); still cast values from the request to the expected type.

Writing

Method	Use
`new myExtItem( $row )` then `store( $fieldFilters = null )`	insert, or update when the key exists; sets the increment key after an insert
`setAttribute( $attr, $val )`	change a value (calls a `set_functions` method if defined)
`remove( $conditions = null, $extraConditions = null )`	delete this row
`removeObject( $def, $conditions = null, $extraConditions = null )` (static)	delete matching rows
`updateObjectList( $parameters )` (static)	update matching rows without fetching them
`sync( $fieldFilters = null )`	store only when there are changes

PHP

```php
<?php
$item = new myExtItem( array( 'parent_id' => 2, 'name' => 'First' ) );
$item->store();
$id = $item->attribute( 'id' );
```

Reading attributes

Method	Returns
`attribute( $attr, $noFunction = false )`	a field's value, or the result of a `function_attributes` method
`hasAttribute( $attr )`	whether the attribute exists
`attributes()`	all attribute names, including function attributes

These three methods make objects readable in templates: `{$item.name}`.

On every engine

`eZPersistentObject` builds the SQL for the configured engine, and on MongoDB installations the adapter turns its calls into MongoDB queries (MongoDB kernel support). It also keeps fields the class does not declare in an internal array, so PHP 8.2's dynamic property deprecation does not apply to subclasses (Dynamic properties).

Related pages

- Previous: eZDB: the database layer
- Next: eZContentObject
- Persistent objects and your own 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.