The debug bar for PHP developers

Using the Exp Debug bar as a PHP developer - the tabs (Messages, Settings, Cache, Timing, SQL, Templates, Included files, Memory, Rendering, User, Server), the summary thresholds, changing debug settings with undo, and...

Source file: `doc/eng-us/help/php-developers/the-debug-bar-for-php-developers.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/the-debug-bar-for-php-developers.md

This page is for PHP developers who use the debug report at the bottom of a page. Since 6.0.15 it is the **Exp Debug bar** (`lib/ezutils/classes/expdebugbarreport.php`): a pinned bar with a live summary of the request and a panel with tabs. The complete reference is The Exp Debug bar; this page covers what a PHP developer uses daily and how to add a panel of your own.

Contents

1. Turning it on
2. The summary
3. The tabs
4. Changing settings from the bar
5. A panel of your own
6. Under Velocity

Turning it on

`site.ini [DebugSettings] DebugOutput=enabled`, limited to your address with `DebugByIP=enabled` and `DebugIPList[]`, on a development or test installation (Debugging with eZDebug). Changing settings and clearing caches from the bar needs the policy `setup/setup` (cache controls: `setup/managecache`).

The summary

The bar shows what `eZDebug` measured for the request: page time, number and time of SQL statements, peak memory and the limit, templates used, messages per level, the engine (Velocity or the SAPI), the siteaccess and, when it is on, the SQL query cache. Each figure gets a level, `ok`, `warn` or `high`, against `Thresholds[]` in `settings/debugbar.ini` (`time_ms`, `sql_count`, `sql_ms`, `memory_mb`, `templates`, as `<warn>;<high>`). A red figure on a page you changed is the first hint of a performance problem.

The tabs

Tab	What a PHP developer reads there
Messages	the notices, warnings and errors your code and the kernel wrote, filterable by level
Settings	every registered debug setting, its value in effect and the file it comes from; change it per scope

Tab	What a PHP developer reads there
Cache	clear this page, content, templates, INI or everything
Timing	timing points and accumulators, yours included (Profiling PHP code)
SQL	number of queries, time, share of the page, and the statements (with `SQLOutput=enabled`), slowest first
Templates	the templates used (`ShowUsedTemplates`) and the packer report
Included files	the PHP files included (`DisplayIncludedFiles`)
Memory	memory now, peak, limit, and the steps that took the most
Rendering	the controller, the content view cache and its key, the compiled templates (for `setup/setup`)
User	the user's cache profile (whether two users share the cached views), preferences, role assignments, and on demand the policies of those roles, by role and by search (for whoever gets the debug output, and `setup/setup`)
Server	where the request entered and which process rendered it (Velocity, PHP-FPM, mod_php, the command line), OPcache status, Velocity's response cache

The filter box above the tabs filters the rows of every tab at once.

Changing settings from the bar

The Settings tab writes through the `exp:ini` engine, in the scope you choose (global override, a siteaccess or an extension). Every write is one JSON line in `var/<site>/log/debugbar.log`, and can be undone from the bar as long as the setting was not changed again since. Debug conditions of `debug.ini [GeneralCondition]` and debug switches of active extensions (a variable whose name contains "debug" with a boolean value) are found automatically and shown as switches.

A panel of your own

An extension can add a tab with data about its part of the request. Register it in `extension/acmereport/settings/debugbar.ini.append.php`:

INI

```
<?php /* #?ini charset="utf-8"?
[DebugBarSettings]
Panels[]=acmereport

[Panel_acmereport]
Class=acmeReportDebugBarPanel
*/ ?>
```

and implement `expDebugBarPanelInterface` (`kernel/classes/debugbar/expdebugbarpanelinterface.php`):

PHP

```
class acmeReportDebugBarPanel implements expDebugBarPanelInterface
{
    public function visible()
    {
        // the bar also shows to anonymous visitors when DebugByIP matches
```

```
        $access = eZUser::currentUser()->hasAccessTo( 'acmereport', 'read' );
        return $access['accessWord'] !== 'no';
    }

    public function title()
    {
        return ezpI18n::tr( 'extension/acmereport', 'Reports' );
    }

    public function chip()
    {
        return array( 'text' => 'Reports: ' . acmeReportStats::count(), 'title' => 'Items read in
 this request', 'level' => 'ok' );
    }

    public function html()
    {
        return '<p>' . htmlspecialchars( acmeReportStats::summary(), ENT_QUOTES | ENT_SUBSTITUTE
| ENT_HTML401, 'UTF-8' ) . '</p>';
    }

    public function data()
    {
        return array( 'items' => acmeReportStats::count() );
    }
}
```

Rules from the bar's reference:

- `visible()` is asked first; a panel that answers no draws nothing.
- `html()` is printed as it is: escape everything you put into it. The chip's text and title are escaped by the bar.
- The class is created for every report, also for anonymous visitors: keep the constructor trivial and the methods cheap, without data access of their own; collect figures while the request runs and only read them here.
- A panel that throws is left out with a warning; the rest of the bar is drawn.
- `data()` goes into the page's JSON under `panels.<id>`.

Under Velocity

The bar names the engine in the Server tab. If a figure looks wrong after a code change, check that the workers were restarted (`exp:velocity deploy`): workers keep the classes they loaded. An engine archive without `expdebugbarreport.php` prints the classic report instead of the bar (Velocity persistent workers for PHP developers).

Related pages

- Exp debug bar (feature)
- Debug bar panel extension point
- Debugging extensions
- 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.