Validating input and casting

How PHP code in Exponential turns untrusted input into safe values - reading through eZHTTPTool and $Params, (int) and (float) casts with ranges, allow lists with strict in_array, string length and character rules...

Source file: doc/eng-us/help/php-developers/validating-input-and-casting.md Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/validating-input-and-casting.md

This page is for PHP developers who take values from requests, settings, imports or the database. The kernel's own way is visible in every view: read through the kernel's helpers, convert the value to the type it must have, check it against what is allowed, and only then use it. Because the code style has no type declarations, these casts are what gives a value its type (Why the code has no type declarations).

Contents

1. Reading input
2. Numbers
3. Choices
4. Text
5. E-mail addresses and URLs
6. Arrays from the request
7. Reporting errors to the user
8. A checklist

Reading input

PHP

```php
$http = eZHTTPTool::instance();
$title = $http->hasPostVariable( 'Title' ) ? (string)$http->postVariable( 'Title' ) : '';
$page  = $http->hasGetVariable( 'page' ) ? (int)$http->getVariable( 'page' ) : 1;
$id    = isset( $Params['ItemID'] ) ? (int)$Params['ItemID'] : 0;
```

Source	Read with
POST	`$http->hasPostVariable()`, `$http->postVariable( $name, $fallback )`
GET	`$http->hasGetVariable()`, `$http->getVariable( $name, $fallback )`
either	`$http->variable( $name, $fallback )`
view parameters	`$Params['Name']` (`params` and `unordered_params` of `module.php`)
form buttons	`$Module->isCurrentAction()`, `$Module->actionParameter()`
uploads	`eZHTTPFile::canFetch()`, `eZHTTPFile::fetch()`

Every one of them returns strings, arrays or `null`. Under Velocity the kernel prepares the

request per worker request; reading `$_POST` and `$_GET` directly bypasses that.

Numbers

PHP

```php
$id     = (int)$Params['ItemID'];                                  // ids
$limit  = max( 1, min( 100, (int)$http->getVariable( 'limit', 20 ) ) );   // bounded
$offset = max( 0, (int)$http->getVariable( 'offset', 0 ) );
$amount = (float)str_replace( ',', '.', (string)$http->postVariable( 'Amount', '0' ) );
```

- `(int)` turns anything into an integer (`'12abc'` becomes 12, `'abc'` becomes 0): check the range after the cast.
- An id of `0` means "not given": treat it as not found.
- Bound every limit: an unbounded `limit` is a denial of service.

Choices

PHP

```php
$sort = (string)$http->getVariable( 'sort', 'created' );
if ( !in_array( $sort, array( 'created', 'title', 'amount' ), true ) )
    $sort = 'created';
```

Always pass `true` as the third argument of `in_array()`: without it, the comparison is loose, so `'1e1'` matches `'10'`, `null` matches `''` and `true` matches any non-empty string. Use allow lists for anything that becomes a column name, a sort direction, a template name, a class name or a file name.

Text

PHP

```php
$title = trim( (string)$http->postVariable( 'Title', '' ) );
if ( $title === '' )
    $errors['title'] = ezpI18n::tr( 'extension/acmereport', 'Enter a title.' );
else if ( mb_strlen( $title, 'UTF-8' ) > 200 )
    $errors['title'] = ezpI18n::tr( 'extension/acmereport', 'The title can have at most 200 chara
cters.' );
```

- Cast to string first (`(string)`), so `null` and arrays never reach string functions (a PHP 8.1 deprecation and a `TypeError` for arrays).
- Count characters with `mb_strlen()`, not bytes.
- Keep text as the user wrote it; escape when you output it, not when you store it.
- For identifiers (a slug, a code), match a pattern: `preg_match( '/^[a-z0-9_-]{1,64}$/', $code )`.

E-mail addresses and URLs

PHP

```php
$email = trim( (string)$http->postVariable( 'Email', '' ) );
$validEmail = filter_var( $email, FILTER_VALIDATE_EMAIL ) !== false;

$url = trim( (string)$http->postVariable( 'Website', '' ) );
$scheme = strtolower( (string)parse_url( $url, PHP_URL_SCHEME ) );
$validUrl = filter_var( $url, FILTER_VALIDATE_URL ) !== false && in_array( $scheme, array( 'http'
, 'https' ), true );
```

Allow only `http` and `https` for URLs you will link or fetch. Redirect targets have their own rules (Security in PHP code).

Arrays from the request

A field can arrive as an array (`Title[]=x`) even when your form sends a string. Check the type:

PHP

```php
$ids = $http->postVariable( 'SelectedIDs', array() );
$ids = is_array( $ids ) ? array_values( array_filter( array_map( 'intval', $ids ) ) ) : array();
```

Reporting errors to the user

Collect errors per field, re-render the form with the values the user typed, and show the errors next to the fields and in a summary at the top. Datatypes report through the kernel's validator instead: `$objectAttribute->setValidationError( ... )` and `return eZInputValidator::STATE_INVALID;` (Write a datatype).

A checklist

- ☐ Every value read through `eZHTTPTool`, `$Params` or the module's actions.
- ☐ Every number cast and bounded.
- ☐ Every choice checked against an allow list with strict comparison.
- ☐ Every string cast, trimmed, length-checked in characters.
- ☐ Every array checked with `is_array()` before use.
- ☐ Errors shown to the user without echoing raw input.

Related pages

- Input validation and output escaping
- eZHTTPTool reference
- Datatype input hardening
- 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.