The request lifecycle

What happens to an HTTP request in Exponential, step by step - index.php, the kernel, siteaccess matching, settings, session, URL alias and wildcard translation, request rules, the module and policy checks, the view…

Source file: `doc/eng-us/help/web-framework-developers/the-request-lifecycle.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/web-framework-developers/the-request-lifecycle.md

This page is for developers who want to know where their code runs on a web request, the way they know Symfony's `HttpKernel` events, Laravel's middleware pipeline or Django's request and response cycle. The authoritative description is The request lifecycle in the kernel architecture; this page walks through it with framework eyes.

Contents

1. The steps
2. The same steps as a pipeline
3. Where you can hook in
4. Under persistent workers
5. Command line requests

The steps

1. `index.php`**.** If `var/maintenance.json` exists, `expMaintenance::check()` may answer (maintenance mode). A request carrying `exp_repair` is handed to `ezpRepairQueue`. Then `autoload.php` loads Composer's autoloader and the generated class maps, `config.env.php` and `config.php` are read, and `$kernel = new ezpKernel( new ezpKernelWeb() ); echo $kernel->run()->getContent();` runs the request.
2. **Kernel construction.** `ezpKernelWeb` reads `i18n.ini` and `site.ini`, activates the global extensions, matches the **siteaccess** with `eZSiteAccess::match()` on the request URI and host, switches to it with `eZSiteAccess::change()`, activates the siteaccess's extensions and reloads the settings so the siteaccess layer applies (Siteaccesses).
3. **Session and request initialization** (`sessionInit()`, `requestInit()`), then the events `request/preinput` and `request/input`. The CSRF check of `ezformtoken` is a `request/input` listener.
4. **URL translation.** A URL alias (`/About-us`) becomes a system address (`content/view/full/58`) through `eZURLAliasML`; URL wildcards are applied through `eZURLWildcard` (Pretty URLs).
5. **Module lookup and access checks.** The module must exist in a module repository and be allowed (`eZModule::accessAllowed()`); the siteaccess login check (`user/login` policy) and the view's policy functions (`eZUser::hasAccessToView()`) run before the view. Request rules decide after the alias translation and after the policies allowed the view (Middleware: request rules).

6. **The view runs.** `$module->run( $functionName, $params, ... )` executes the view script or `ModuleView` class, which returns a `$Result` array. A view can ask for a rerun with another URI or a redirect (`eZModule::STATUS_RERUN`, `STATUS_REDIRECT`); the dispatch loop handles both (Controllers).
7. **Page layout and output.** The result's `content` is placed into the siteaccess's `pagelayout.tpl` (unless the view chose another or none), passed through the output filters `response/preoutput` and `response/output`, and returned. `ezformtoken` adds its hidden token to forms in a `response/output` listener.
8. **Shutdown.** Pending data is written; under Velocity the worker resets its state for the next request.

The same steps as a pipeline

Concern	Symfony	Laravel	Django	Exponential
Entry	`public/index.php`, `Kernel::handle()`	`public/index.php`, HTTP kernel	WSGI or ASGI handler	`index.php`, `ezpKernel::run()`
Early answer	`kernel.request` listener	global middleware	middleware	maintenance check, HTTP cache early exit, Velocity response cache
Tenant or site	host requirement, custom listener	domain routes	`Site` framework	siteaccess matching
Routing	router	router	URLconf	URL alias and wildcard translation, then `<module>/<view>`
Authorization before the action	firewall, voters	`auth` and `can` middleware	decorators	view `functions`, siteaccess login check, request rules
Action	controller	controller	view function	module view
Rendering	Twig	Blade	templates	template language and pagelayout
Response filter	`kernel.response`	terminable or after middleware	middleware	`response/output` filter

Where you can hook in

Need	Hook	Read
Answer or refuse before routing	`request/preinput` or `request/input` listener	Events and listeners
Deny, redirect, rewrite or log by rule after the alias is known	request rules	Middleware
A new address	a module and a view in your extension	Routing
Change a kernel view	a view class re-implementation (`site.ini [RunnableSettings] Implementation[]`) or a module override	Controllers

Need	Hook	Read
Change the final HTML	`response/output` filter listener	Events and listeners
Run around content operations	workflows and triggers	Workflows

Under persistent workers

Under Velocity the application is loaded once in a parent process and forked into workers that each serve many requests. Steps 2 to 8 run per request, but anything a class keeps in a static property survives into the next request unless it is reset. The kernel's rules: no per-request state in statics without a reset, no `exit` for flow control except through `eZExecution::cleanExit()`, and a changed class or setting reaches the workers only after `./console exp:velocity restart` or `exp:velocity deploy` (Servers: Velocity, PHP-FPM and FrankenPHP). If you know Laravel Octane or Symfony on RoadRunner, the discipline is the same.

Command line requests

Console commands, cronjob parts and queue workers do not use `ezpKernelWeb`. They boot through `eZScript`, which loads settings for the siteaccess given with `-s`, and run as the anonymous user unless the code logs a user in (Command line).

Related pages

- Architecture for application creators
- Front controller pattern
- Velocity: the persistent-worker server
- 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.