Modul-Views als Klassen

Wie Modul-Views, Befehle und Cronjob-Teile seit Exponential 6.0.15 als Klassen geschrieben werden - die Runnable-Hierarchie, die einzeiligen Einstiegsdateien, der Scope, den eine View erhält, Dienste, die sich Views und…

Source file: `doc/ger-de/hilfe/php-entwickler/modul-views-als-klassen.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/module-views-as-classes.md

Diese Seite richtet sich an PHP-Entwickler, die Views, Befehle oder Cronjob-Teile schreiben oder die des Kernels ändern. Ab Exponential 6.0.15 liegt der Code jeder Kernel-View, jedes Befehls und jedes Cronjob-Teils in einer Klasse; die Datei, die Sie kennen, behält ihren Pfad und ruft diese Klasse auf. Die vollständige Referenz ist Befehle, Cronjob-Teile und Modul-Views als Klassen.

Inhalt

1. Warum Klassen
2. Die Hierarchie
3. Wo die Klassen liegen
4. Der Scope, den eine View erhält
5. Dienste: eine Implementierung für drei Einstiegspunkte
6. Ereignisse rund um run()
7. Eine Kernel-View neu implementieren
8. Ein Runnable aus Code aufrufen
9. Der alte Weg funktioniert weiterhin

Warum Klassen

Eine einfache View-Datei lässt sich nicht erweitern, nicht für sich aufrufen und nicht für sich testen. Eine Klasse schon: Eine Site oder eine Erweiterung kann eine Methode einer Kernel-View über eine Einstellung ersetzen, ein Befehl und eine View können sich einen Dienst teilen, und ein Test kann die Klasse ausführen. Deshalb wird neuer Code als Klassen geschrieben, und deshalb wurde jede Kernel-View, jeder Befehl und jeder Cronjob-Teil in eine verschoben.

Die Hierarchie

```
Exponential\Runnable\Runnable           kernel/private/classes/runnable/runnable.php
  Exponential\Runnable\Command          a command line script      run()
  Exponential\Runnable\CronjobPart      a cronjob part             run( array $scope )
  Exponential\Runnable\ModuleView       a module view              run( array $scope )
```

`ModuleView::main( $scriptFile, array $scope )` erzeugt die Instanz (`Runnable::create()`, das Neuimplementierungen berücksichtigt), führt `run()` zwischen den Ereignissen aus und gibt das Ergebnis der View zurück: `$Result`, wenn die View es gesetzt hat, sonst das, was `run()` zurückgegeben hat (`ModuleView::viewResult()`).

Wo die Klassen liegen

Art	Einstiegsdatei	Klasse
Kernel-View	`kernel/<module>/<view>.php`	`Exponential\View\Kernel\<Module>\<View>` in `kernel/private/classes/views/<module>/<view>.php`
Kernel-Befehl	`bin/php/<name>.php`	`Exponential\Command\Kernel\<Name>` in `kernel/private/classes/commands/<name>.php`
Kernel-Cronjob-Teil	`cronjobs/<name>.php`	`Exponential\Cronjob\Kernel\<Name>` in `kernel/private/classes/cronjobs/<name>.php`
View einer Erweiterung	`extension/<ext>/modules/<module>/<view>.php`	`Exponential\View\Extension\<Ext>\<Module>\<View>` in `extension/<ext>/classes/runnable/views/<module>/`
Befehl einer Erweiterung	`extension/<ext>/bin/php/<name>.php`	`Exponential\Command\Extension\<Ext>\<Name>` in `extension/<ext>/classes/runnable/commands/`
Cronjob-Teil einer Erweiterung	`extension/<ext>/cronjobs/<name>.php`	`Exponential\Cronjob\Extension\<Ext>\<Name>` in `extension/<ext>/classes/runnable/cronjobs/`

Eine View, deren Name ein reserviertes Wort in PHP ist, bekommt ein Suffix: `kernel/section/list.php` ruft `Exponential\View\Kernel\Section\ListView` auf.

Der Scope, den eine View erhält

Die Einstiegsdatei übergibt alles, was `eZProcess::runFile()` definiert hat:

PHP

```php
<?php
return \Exponential\View\Kernel\Content\History::main( __FILE__, get_defined_vars() );
```

In `$scope`	Enthält
`$scope['Params']`	die Parameter der View; das Modul ist `$scope['Params']['Module']`
`$scope['Result']`	die Variable `$Result`, zu Beginn leer
die Parameter der View als eigene Schlüssel	nur wenn `module.php` `'variable_params' => true` setzt

Lesen Sie das Modul als `$scope['Params']['Module']`, das jedes Modul hat. Innerhalb der Klasse liefern `$this->scriptFile()` und `$this->scriptDir()`, was `__FILE__` und `__DIR__` in der Einstiegsdatei bedeuteten. Klassennamen innerhalb eines Namespace werden voll qualifiziert geschrieben (`\eZTemplate`, `\eZINI`).

Dienste: eine Implementierung für drei Einstiegspunkte

Wo eine View und ein Befehl oder Cronjob-Teil dieselbe Arbeit erledigen, liegt die Arbeit in einer Dienstklasse, die alle aufrufen. Beispiele aus dem Kernel:

Arbeit	Dienst	Aufgerufen von
den Papierkorb leeren	`Exponential\Service\Trash`	`content/trash`, `bin/php/trashpurge.php`, `cronjobs/trashpurge.php`
abgelaufene Sitzungen entfernen	`Exponential\Service\SessionGarbageCollector`	`setup/session`, `bin/php/ezsessiongc.php`, `cronjobs/session_gc.php`
Caches leeren	`expCacheManager`	`setup/cache`, `bin/php/cache.php` (`exp:cache`), die Cache-Werkzeugleiste
alte Entwürfe entfernen	`Exponential\Service\DraftsCleanup`	`cronjobs/old_drafts_cleanup.php`, `cronjobs/internal_drafts_cleanup.php`

Machen Sie es in Ihrer Erweiterung genauso: Legen Sie die Arbeit in eine einfache Klasse mit Werten hinein und Werten heraus, und halten Sie View, Befehl und Cronjob-Teil schlank. Der Dienst ist der Teil, den Sie testen (PHPUnit ohne Datenbank).

Ereignisse rund um run()

`main()` meldet `runnable/<kind>/before` (mit `$runnable, $class, $scope`) und filtert das Ergebnis durch `runnable/<kind>/after` (mit `$result, $runnable, $class, $scope`), wobei `<kind>` `command`, `cronjob` oder `view` ist. Registrieren Sie Listener in `site.ini`:

INI

```
[RunnableSettings]
Listeners[]=runnable/view/before@acmeViewTimer::before
Listeners[]=runnable/view/after@acmeViewTimer::after
```

Ein `after`-Listener ist ein Filter: Er muss das Ergebnis zurückgeben, sonst verliert die View es. `[RunnableSettings] Listeners[]` werden in jedem Kontext angehängt, auch auf der Kommandozeile; `[Event] Listeners[]` nur für Webanfragen (Ereignisse, Listener und Filter).

Eine Kernel-View neu implementieren

Statt eine View-Datei des Kernels zu kopieren, leiten Sie von ihrer Klasse ab und registrieren die Unterklasse:

INI

```
[RunnableSettings]
Implementation[Exponential\View\Kernel\Content\History]=acmeHistoryView
```

PHP

```
class acmeHistoryView extends \Exponential\View\Kernel\Content\History
{
    public function run( array $scope )
    {
        $result = parent::run( $scope );
        // change $result['content'] or $result['path'] here
        return $result;
    }
}
```

Der Ersatz muss die Klasse erweitern, die er ersetzt; ein Eintrag, der etwas anderes nennt, wird ignoriert. Keine Kernel-Datei wird kopiert, sodass das nächste Release das Original unter Ihrer Unterklasse aktualisiert. Für Befehle werden `Implementation[]` und `Listeners[]` nur aus

`settings/site.ini` und `settings/override/site.ini.append.php` gelesen, weil ein Befehl erzeugt wird, bevor sein Skript die übrigen Einstellungen liest. **Setup > RAD** und seine Übersicht listen jedes Runnable und jeden `Implementation[]`-Eintrag auf, der nicht funktionieren kann.

Ein Runnable aus Code aufrufen

PHP

```php
$result = \Exponential\View\Kernel\Content\History::create( 'kernel/content/history.php' )
            ->run( array( 'Params' => $params, 'Module' => $module ) );
```

`create()` berücksichtigt `Implementation[]`; ein direkter Aufruf von `run()` löst die Ereignisse nicht aus (`main()` tut es).

Der alte Weg funktioniert weiterhin

Eine View, ein Befehl oder ein Cronjob-Teil, der als einfache PHP-Datei geschrieben ist, läuft genau wie bisher; nichts verlangt die Klassenform. Schreiben Sie neuen Code als Klassen, und verschieben Sie alten Code, wenn Sie ihn ohnehin ändern.

Verwandte Seiten

- Schreiben Sie Ihre erste Modul-View
- Einen Konsolenbefehl schreiben
- Einen Cronjob-Teil schreiben
- Freie Dokumentation: diese Seite darf unter der GNU FDL 1.3 frei kopiert, übersetzt und verbessert werden.

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

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.