

Warming and Commands

Warming and Commands

Anonymous · 10/10/2026

Warming and Commands

CacheWarmer and WarmableInterface: Warming Without a Kernel

```
use Derafu\Cache\CacheWarmer;
use Derafu\Cache\Contract\WarmableInterface;

final class ExampleWarmable implements WarmableInterface
{
    public function warmup(): void { /* ... */ }
}

$failures = (new CacheWarmer())->warmup([$warmableA, $warmableB]);
// list<array{warmable: WarmableInterface, exception: Throwable}> – empty if
everything succeeded.
```

`symfony/cache` has no warmup concept at all — only `PruneableInterface` (the opposite: removing *expired* entries). Symfony's real `CacheWarmerInterface` lives in `symfony/http-kernel`, tied to the full framework Kernel's boot lifecycle: it warms the *framework's own* caches (compiled container, routing) into a temporary location and atomically swaps them in, specifically because a half-written container cache would take the whole application down.

Warming an application-level PSR-6 cache doesn't carry that risk — if a warmable fails partway through, the next real read just recomputes that one entry. So `CacheWarmer` doesn't try to replicate the atomic swap or the Kernel's boot-order resolution: it runs each warmable in order, keeps going if one throws, and reports every failure at the end instead of stopping at the first one.

There's deliberately no bundled warmup *command*: a generic one would need to discover which `WarmableInterface` instances exist in a given application and how to construct each one — almost none will have a no-argument constructor, since a real warmable needs whatever it's warming. That discovery is exactly what a DI container is for, which this package chooses not to depend on (see [Console Commands](#) below). The intended shape is a small, application-specific command that constructs the warmables it actually has and calls `(new CacheWarmer())->warmup($warmables)`.

Console Commands (Optional)

```
use Derafu\Cache\Console\ClearLocalCacheCommand;  
use Derafu\Cache\Console\LocalCacheStatsCommand;  
  
$application->add(new ClearLocalCacheCommand());  
$application->add(new LocalCacheStatsCommand());
```

```
php bin/console derafu:local-cache:clear /var/cache/my_package  
php bin/console derafu:local-cache:stats /var/cache/my_package
```

Both wrap `CacheDirectory` and require only a path. Named `derafu:local-cache:*`, not the more obvious `cache:clear/cache:stats` — that name is already taken in any real Symfony full-stack app (`symfony/framework-bundle` ships its own `cache:clear`, for the framework's own internal cache), and a generic name from a library meant to be added into someone else's `Application` is a collision waiting to happen. Still want a shorter name? Rename after construction: `(new ClearLocalCacheCommand())->setName('cache:clear-local')`.

Neither command is registered automatically — this package has no DI container of its own to register commands into. `symfony/console` is a suggest, not a hard requirement: Composer's autoloading is lazy, so a consumer that never references `Console\*` pays nothing for it even without `symfony/console` installed.

Last updated on 10/10/2026