

The symfony Reference Guide

symfony 1.2

Fabien Potencier

The symfony Reference Guide | symfony 1.2 | version *reference-1.2-en-2009-05-25*

© 2009 Fabien Potencier

ISBN-13: 978-2-918390-05-3

Editor: Fabien Potencier

Proofreader: Kris Wallsmith

Cover Design: Franck Bodiot

Indexer: Fabien Potencier

Icons: DocBook XSL stylesheets

Sensio SA

92-98, boulevard Victor Hugo

92 115 Clichy

France

info@sensio.com

This work is licensed under the “Attribution-Share Alike 3.0 Unported” license (<http://creativecommons.org/licenses/by-sa/3.0/>).

You are free **to share** (to copy, distribute and transmit the work), and **to remix** (to adapt the work) under the following conditions:

- **Attribution:** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Share Alike:** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license. For any reuse or distribution, you must make clear to others the license terms of this work.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Sensio shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

If you find typos or errors, feel free to report them by creating a ticket on the symfony ticketing system (<http://trac.symfony-project.org/register>). Based on tickets and users feedback, this book is continuously updated.

You can contact the author about this book, symfony and Open-Source at fabien.potencier@symfony-project.com or for training, consulting, application development, or business related questions at fabien.potencier@sensio.com.

The settings.yml Configuration File

Most aspects of symfony can be configured either via a configuration file written in YAML, or with plain PHP. In this section, the main configuration file for an application, `settings.yml`, will be described.

The main `settings.yml` configuration file for an application can be found in the `apps/APP_NAME/config/` directory.

As discussed in the introduction, the `settings.yml` file is **environment-aware** (*page 30*), and benefits from the **configuration cascade mechanism** (*page 30*).

Each environment section has two sub-sections: `.actions` and `.settings`. All configuration directives go under the `.settings` sub-section, except for the default actions to be rendered for some common pages.



The `settings.yml` configuration file is cached as a PHP file; the process is automatically managed by the `sfDefineEnvironmentConfigHandler` class (*page 111*).

Settings

- `.actions`
 - `error_404` (page 35)
 - `login` (page 35)
 - `secure` (page 35)
 - `module_disabled` (page 35)
- `.settings`
 - `cache` (page 38)
 - `charset` (page 37)
 - `check_lock` (page 41)
 - `check_symfony_version` (page 41)
 - `compressed` (page 40)
 - `csrf_secret` (page 37)
 - `default_culture` (page 38)
 - `default_timezone` (page 37)
 - `enabled_modules` (page 37)
 - `error_reporting` (page 40)
 - `escaping_strategy` (page 36)
 - `escaping_method` (page 36)
 - `etag` (page 38)
 - `i18n` (page 38)
 - `lazy_cache_key` (page 39)
 - `logging_enabled` (page 39)
 - `no_script_name` (page 39)
 - `max_forwards` (page 42)
 - `standard_helpers` (page 39)
 - `strip_comments` (page 41)
 - `use_database` (page 41)
 - `web_debug` (page 40)
 - `web_debug_web_dir` (page 41)

The .actions Sub-Section

Default configuration:

```
default:
  .actions:
    error_404_module:      default
    error_404_action:     error404

    login_module:         default
    login_action:         login

    secure_module:        default
    secure_action:        secure

    module_disabled_module: default
    module_disabled_action: disabled
```

*Listing
4-1*

The `.actions` sub-section defines the action to execute when common pages must be rendered. Each definition has two components: one for the module (suffixed by `_module`), and one for the action (suffixed by `_action`).

`error_404`

The `error_404` action is executed when a 404 page must be rendered.

`login`

The `login` action is executed when a non-authenticated user tries to access a secure page.

`secure`

The `secure` action is executed when a user doesn't have the required credentials.

`module_disabled`

The `module_disabled` action is executed when a user requests a disabled module.

The .settings Sub-Section

The `.settings` sub-section is where the framework configuration occurs. The paragraphs below describe all possible settings and are roughly ordered by importance.

All settings defined in the `.settings` section are available anywhere in the code by using the `sfConfig` object and prefixing the setting with `sf_`. For instance, to get the value of the `charset` setting, use:

Listing
4-2 `sfConfig::get('sf_charset');`

`escaping_strategy`

Default: off

The `escaping_strategy` setting is a Boolean setting that determines if the output escaper sub-framework is enabled. When enabled, all variables made available in the templates are automatically escaped by calling the helper function defined by the `escaping_method` setting (see below).

Be careful that the `escaping_method` is the default helper used by symfony, but this can be overridden on a case by case basis, when outputting a variable in a JavaScript script tag for example.

The output escaper sub-framework uses the `charset` setting for the escaping.

It is highly recommended to change the default value to `on`.



This settings can be set when you create an application with the `generate:app` task by using the `--escaping-strategy` option.

`escaping_method`

Default: ESC_SPECIALCHARS

The `escaping_method` defines the default function to use for escaping variables in templates (see the `escaping_strategy` setting above).

You can choose one of the built-in values: `ESC_SPECIALCHARS`, `ESC_RAW`, `ESC_ENTITIES`, `ESC_JS`, `ESC_JS_NO_ENTITIES`, and `ESC_SPECIALCHARS`, or create your own function.

Most of the time, the default value is fine. The `ESC_ENTITIES` helper can also be used, especially if you are only working with English or European languages.

csrf_secret

Default: false

The `csrf_secret` is a unique secret for your application. If not set to `false`, it enables CSRF protection for all forms defined with the form framework. This settings is also used by the `link_to()` helper when it needs to convert a link to a form (to simulate a DELETE HTTP method for example).

It is highly recommended to change the default value to a unique secret.



This settings can be set when you create an application with the `generate:app` task by using the `--csrf-secret` option.

charset

Default: utf-8

The `charset` setting is the charset that will be used everywhere in the framework: from the response Content-Type header, to the output escaping feature.

Most of the time, the default is fine.

enabled_modules

Default: [default]

The `enabled_modules` is an array of module names to enable for this application. Modules defined in plugins or in the symfony core are not enabled by default, and must be listed in this setting to be accessible.

Adding a module is as simple as appending it to the list (the order of the modules do not matter):

```
enabled_modules: [default, sfGuardAuth]
```

Listing
4-3

The `default` module defined in the framework contains all the default actions set in the `.actions` sub-section of `settings.yml`. It is recommended that you customize all of them, and then remove the `default` module from this setting.

default_timezone

Default: none

The `default_timezone` setting defines the default timezone used by PHP. It can be any timezone³ recognized by PHP.



If you don't define a timezone, you are advised to define one in the `php.ini` file. If not, symfony will try to guess the best timezone by calling the `date_default_timezone_get()`⁴ PHP function.

cache

Default: off

The `cache` setting enables or disables template caching.



The general configuration of the cache system is done in the `view_cache_manager` (page 51) and `view_cache` (page 52) sections of the `factories.yml` configuration file. The fined-grained configuration is done in the `cache.yml` (page 85) configuration file.

etag

Default: on by default except for the dev and test environments

The `etag` setting enables or disables the automatic generation of ETag HTTP headers. The ETag generated by symfony is a simple md5 of the response content.

i18n

Default: off

The `i18n` setting is a Boolean that enables or disables the i18n sub-framework. If your application is internationalized, set it to `on`.



The general configuration of the i18n system is to be done in the `i18n` (page 52) section of the `factories.yml` configuration file.

default_culture

Default: en

The `default_culture` setting defines the default culture used by the i18n sub-framework. It can be any valid culture.

3. <http://www.php.net/manual/en/class.datetimezone.php>

4. http://www.php.net/date_default_timezone_get

standard_helpers

Default: [Partial, Cache, Form]

The `standard_helpers` setting is an array of helper groups to load for all templates (name of the group helper without the `Helper` suffix).

no_script_name

Default: `on` for the `prod` environment of the first application created, `off` for all others

The `no_script_name` setting determines whether the front controller script name is prepended to generated URLs or not. By default, it is set to `on` by the `generate:app` task for the `prod` environment of the first application created.

Obviously, only one application and environment can have this setting set to `on` if all front controllers are in the same directory (`web/`). If you want more than one application with `no_script_name` set to `on`, move the corresponding front controller(s) under a sub-directory of the web root directory.

lazy_cache_key

Default: `on` for new projects, `off` for upgraded projects

When enabled, the `lazy_cache_key` setting delays the creation of a cache key until after checking whether an action or partial is cacheable. This can result in a big performance improvement, depending on your usage of template partials.

This setting was introduced in symfony 1.2.7 to improve performance without breaking backward compatibility with previous 1.2 releases. It will be removed in symfony 1.3 as the optimization will always be enabled.



This setting is only available for symfony 1.2.7 and up.

logging_enabled

Default: `on` for all environments except `prod`

The `logging_enabled` setting enables the logging sub-framework. Setting it to `false` bypasses the logging mechanism completely and provides a small performance gain.



The fined-grained configuration of the logging is to be done in the `factories.yml` configuration file.

web_debug

Default: `off` for all environments except `dev`

The `web_debug` setting enables the web debug toolbar. The web debug toolbar is injected into a page when the response content type is HTML.

error_reporting

Default:

- `prod:` `E_PARSE | E_COMPILE_ERROR | E_ERROR | E_CORE_ERROR | E_USER_ERROR`
- `dev:` `E_ALL | E_STRICT`
- `test:` `(E_ALL | E_STRICT) ^ E_NOTICE`
- `default:` `E_PARSE | E_COMPILE_ERROR | E_ERROR | E_CORE_ERROR | E_USER_ERROR`

The `error_reporting` setting controls the level of PHP error reporting (to be displayed in the browser and written to the logs).



The PHP website has some information about how to use bitwise operators⁵.

The default configuration is the most sensible one, and should not be altered.



The display of errors in the browser is automatically disabled for front controllers that have `debug` disabled, which is the case by default for the `prod` environment.

compressed

Default: `off`

The `compressed` setting enables native PHP response compression. If set to `on`, symfony will use `ob_gzhandler`⁶ as a callback function for `ob_start()`.

It is recommended to keep it to `off`, and use the native compression mechanism of your web server instead.

5. <http://www.php.net/language.operators.bitwise>

6. http://www.php.net/ob_gzhandler

use_database

Default: on

The `use_database` determines if the application uses a database or not.

check_lock

Default: off

The `check_lock` setting enables or disables the application lock system triggered by some tasks like `cache:clear` and `project:disable`.

If set to `on`, all requests to disabled applications are automatically redirected to the symfony core `lib/exception/data/unavailable.php` page.



You can override the default unavailable template by adding a `config/unavailable.php` file to your project or application.

check_symfony_version

Default: off

The `check_symfony_version` enables or disables checking of the current symfony version upon every request. If enabled, symfony will clear the cache automatically when the framework is upgraded.

It is highly recommended to not set this to `on` as it adds a small overhead, and because it is simple to just clear the cache when you deploy a new version of your project. This setting is only useful if several projects share the same symfony code, which is not recommended.

web_debug_web_dir

Default: `/sf/sf_web_debug`

The `web_debug_web_dir` sets the web path to the web debug toolbar assets (images, stylesheets, and JavaScript files).

strip_comments

Default: on

The `strip_comments` determines if symfony should strip the comments when compiling core classes. This setting is only used if the application `debug` setting is set to `off`.

If you have blank pages in the production environment only, try to set this setting to off.

`max_forwards`

Default: 5

The `max_forwards` setting sets the maximum number of internal forwards allowed before symfony throws an exception. This is to avoid infinite loops.