



The settings.yml Configuration File

This PDF is brought to you by
SENSIOLABS 

License: Creative Commons Attribution-Share Alike 3.0 Unported License
Version: 01-Settings-1.2-en-2009-02-16

Table of Contents

Chapter 1: The settings.yml Configuration File	3
Introduction	3
Settings	4
The .actions Sub-Section	5
error_404	5
login	5
secure	5
module_disabled	5
The .settings Sub-Section	5
escaping_strategy	5
escaping_method	6
csrf_secret	6
charset	6
enabled_modules	6
cache	7
etag	7
i18n	7
default_culture	7
standard_helpers	7
no_script_name	7
logging_enabled	8
web_debug	8
error_reporting	8
compressed	8
use_database	9
check_lock	9
check_symfony_version	9
web_debug_web_dir	9
strip_comments	9
max_forwards	9

The settings.yml Configuration File

Introduction

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 `settings.yml` file is to be found in the `apps/APP_NAME/config/` directory.

The file is environment aware, which means that there is a section for each one of them. When creating a new application, symfony creates sensible configuration for the three default symfony environments: `prod`, `test`, and `dev`:

```
prod:
  # Configuration for the `prod` environment
test:
  # Configuration for the `test` environment
dev:
  # Configuration for the `dev` environment
all:
  # Default configuration for all environments
```

*Listing
1-1*

The `settings.yml` file from an application inherits from the configuration set in the main `config/` directory of the project, and eventually from the default configuration contained in the framework itself (`lib/config/config/settings.yml`).

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.

Settings

- [cache \(page 7\)](#)
- [charset \(page 6\)](#)
- [check_lock \(page 9\)](#)
- [check_symfony_version \(page 9\)](#)
- [compressed \(page 8\)](#)
- [csrf_secret \(page 6\)](#)
- [default_culture \(page 7\)](#)
- [enabled_modules \(page 6\)](#)
- [error_reporting \(page 8\)](#)
- [escaping_strategy \(page 5\)](#)
- [escaping_method \(page 6\)](#)
- [etag \(page 7\)](#)
- [i18n \(page 7\)](#)
- [logging_enabled \(page 8\)](#)
- [no_script_name \(page 7\)](#)
- [max_forwards \(page 9\)](#)
- [standard_helpers \(page 7\)](#)
- [strip_comments \(page 9\)](#)
- [use_database \(page 9\)](#)
- [web_debug \(page 8\)](#)
- [web_debug_web_dir \(page 9\)](#)

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
1-2*

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 all 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:

```
sfConfig::get('sf_charset');
```

*Listing
1-3*

escaping_strategy

Default: off

The `escaping_strategy` setting is a Boolean setting that determines if the output escaper sub-framework must be 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
1-4

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.

cache

Default: off

The `cache` setting enables or disables template caching.



The general configuration of the cache system is to be done in the `factories.yml` configuration file. The fined-grained configuration is to be done in the `cache.yml` configuration files.

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`.

default_culture

Default: en

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

standard_helpers

Default: [Partial, Cache, Form]

The `standard_helpers` 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 created application, off for all others

The `no_script_name` setting determines if the front controller script name must be added before the generated URLs or not. By default, it is set to `on` by the `generate:app` task for the `prod` environment of the very 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.

logging_enabled

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

The `logging_enabled` setting enables the logging sub-framework.



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 added to pages with a response content type of `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 operators¹.

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



The display of errors in the browser is automatically disabled for applications 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`² 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.

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

2. 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 the `cache:clear` one.

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

check_symfony_version

Default: off

The `check_symfony_version` enables or disables symfony version check for every request.

If enables, 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 forwarding allowed before symfony throws an exception. This is to avoid infinite loops.