Directory Structure and CLI (Tasks)

DIRECTORY STRUCTURE

```
myproject (project root)
  apps
    myapp
      config
        app.yml
        cache.yml
        databases.yml
        factories.yml
        filters.yml
        i18n.yml
        myappConfiguration.class.php
        routing.yml
        security.yml
        settings.yml
        view.yml
      i18n
      lib
      modules
        mymodule
          actions
            actions.class.php
            components.class.php
          i18n
          config
          lib
          templates
            indexSuccess.php
          validate
      templates
        layout.php
    batch
    cache
      myapp
      tmp
    config
      databases.yml
      propel.ini
      properties.ini
      ProjectConfiguration.class.php
      schema.yml (schema.xml)
      rsync_exclude.txt
      doctrine
        schema.yml
    data
      fixtures
      model
      sql
    lib
      model
        map
        om
        doctrine
      filter
      form
    log
      myapp_dev.log
      myapp_prod.log
    plugins
    test
      bootstrap
      functional
      unit
    web
      css
      images
      js
      uploads
        assets
      index.php
      myapp_dev.php
      robots.txt
      .htaccess
    symfony
```

COMMAND LINE INTERFACE (CLI) - TASKS

Usage: `symfony [options] task_name [arguments]`

Default options and arguments for every symfony task:

`--help`	`-H`	Display this help message
`--quiet`	`-q`	Do not log messages to standard output
`--trace`	`-t`	Turn on invoke/execute tracing, enable full backtrace
`--version`	`-V`	Display the program version
`--color`		Forces ANSI color output
`--xml`		To output help as XML

A **task name** can be composed of an optional namespace (e.g: generate, project, ...) and a name, separated by a colon (:)

```
$ symfony
```
Full list of the available tasks

```
$ symfony list [--xml] [namespace]
```
List tasks

```
$ symfony -V
```
Installed version of the symfony package and path of the symfony libraries used by CLI

```
$ symfony help [--xml] [task_name]
```
Displays help for a task

```
$ symfony cache:clear [--app=app_name] [--env=[prod|dev]] [--type=type]
```
Clear the cached information (shortcut: cc)
The built-in types are: config, i18n, routing, module and template

```
$ symfony configure:author <author_name>
```
Configures the author for a project. Used by the generates to pre-configure the PHPDoc header

```
$ symfony configure:database [--env=env_name] [--name=conn_name]
  [--class=db_class_name] [--app=app_name] <dsn> [username] [password]
```
Configures the database DSN for a project

```
$ symfony app:routes <app_name> [route_name]
```
Displays current routes for an application

generate

```
$ symfony generate:task [--dir=[lib/task]] [--use-database=
  [doctrine|propel|main|false]] [--brief-description="..."] <task_name>
```
Creates a skeleton class for a new task

```
$ symfony generate:project [--orm=[Doctrine|Propel|none]]
  [--installer=installer_script] <project_name> [author_name]
```
Generates a new project. Creates basic directory structure for a new project in the current directory

```
$ symfony generate:app [--escaping-strategy=...] [--csrf-secret=...] <app_name>
```
Generates a new application. Creates basic directory structure for a new application in the current project. Also creates two front controller scripts in the web/ directory: <app_name>.php for the prod environment and <app_name>_dev.php for the dev.
For the first application, the production environment script is named index.php

```
$ symfony generate:module <app_name> <module_name>
```
Generates a new module. Creates the basic directory structure for a new module in an existing application.

project

```
$ symfony project:clear-controllers
```
Clears all non production environment controllers

```
$ symfony project:deploy [--go] [--rsync-dir=config]
[--rsync-options=[-azC|--force|--delete|--progress]] <server_name>
```
Deploys a project on a server. The server must be configured in config/properties.ini
The --rsync-dir is the directory where to look for rsync*.txt files

```
$ symfony project:disable <env_name> [app1] ... [appN]
```
Disables an application in a given environment

```
$ symfony project:enable <env_name> [app1] ... [appN]
```
Enables an application in a given environment

```
$ symfony project:optimize <app_name> [env_name]
```
Optimizes a project for better performance. This task should only be used on a prod server

```
$ symfony project:permissions
```
Fixes directory permissions

```
$ symfony project:send-emails [--application=app_name] [--env=env_name]
[--message-limit=max_number_msg_to_send] [--time-limit=time_limit_in_sec]
```
Sends emails stored in a queue

```
$ symfony project:validate
```
Detects deprecated usage in your project

CLI (Tasks) - Database - Doctrine ORM

DOCTRINE TASKS

```
$ symfony doctrine:build [--application=app_name] [--env=env_name] [--no-confirmation] [--all] [--all-classes]
[--model] [--forms] [--filters] [--sql] [--db] [--and-migrate] [--and-load=fixture_file] [--and-append=fixture_file]
```
Generates code based on your schema

```
$ symfony doctrine:build-db [--application=app_name] [--env=env_name] [database1] ... [databaseN]
```
Creates one or more databases based on configuration in config/databases.yml

```
$ symfony doctrine:build-filters [--application=app_name] [--env=env_name] [--model-dir-name=...]
[--filter-dir-name=form_dir_name] [--generator-class=generator_class_name]
```
Creates form filter classes from the schema. The classes are created in lib/doctrine/filter. Never overrides custom classes in lib/doctrine/filter. It only replaces base classes generated in lib/doctrine/filter/base.

```
$ symfony doctrine:build-forms [--application=app_name] [--env=env_name] [--model-dir-name=...]
[--form-dir-name=form_dir_name] [--generator-class=generator_class_name]
```
Creates form classes from the schema for the current model. The classes are created in lib/doctrine/form. This task never overrides custom classes in lib/doctrine/form. It only replaces base classes generated in lib/doctrine/form/base

```
$ symfony doctrine:build-model [--application=app_name] [--env=env_name]
```
Creates model classes from the schema. Read the schema information in config/doctrine/*.yml from the project and all enabled plugins. The model classes files are created in lib/model/doctrine. This task never overrides custom classes in lib/model/doctrine. It only replaces files in lib/model/doctrine/base.

```
$ symfony doctrine:build-schema [--application=app_name] [--env=env_name]
```
Creates a schema from an existing database. The task creates a yml file in config/doctrine

```
$ symfony doctrine:build-sql [--application=app_name] [--env=env_name]
```
Creates SQL for the current model. The generated SQL is optimized for the database configured in config/databases.yml

```
$ symfony doctrine:clean-model-files [--no-confirmation]
```
Delete all generated model classes for models which no longer exist in your YAML schema. Alias: doctrine:clean

```
$ symfony doctrine:create-model-tables [--application=app_name] [--env=env_name] [models1] ... [modelsN]
```
Drop and recreate tables for specified models

```
$ symfony doctrine:data-dump [--application=app_name] [--env=env_name] [target_filename]
```
Dumps the database data in data/fixtures/%target_filename%. The dump file is in the YML format and can be reimported by using the doctrine:dataload task

```
$ symfony doctrine:data-load [--application=app_name] [--env=env_name] [--append] [dir_or_file1] ... [dir_or_fileN]
```
Loads data from all the files found in data/fixtures/. Using the option --append, the task don't delete current data in the database

```
$ symfony doctrine:delete-model-files [--no-confirmation] model_name1 ... [model_nameN]
```
Deletes all files associated with certain models

```
$ symfony doctrine:dql [--application=app_name] [--env=env_name] [--show-sql] [--table] <dql_query> [param1] ... [paramN]
```
Executes a DQL query and displays the formatted results
E.g.: $ symfony doctrine:dql "FROM User WHERE email LIKE ?" "%symfony-project.com"

```
$ symfony doctrine:drop-db [--application=app_name] [--env=env_name] [--no-confirmation] [db1] ... [dbN]
```
Drops database for current model

```
$ symfony doctrine:generate-admin [--module=module_name] [--theme=theme_name] [--singular=singular_name] [--plural=plural_name]
[--env=env_name] [--actions-base-class=base_class_for_actions] <app_name> <route_or_model>
```
Generates a Doctrine admin module

```
$ symfony doctrine:generate-migration [--application=app_name] [--env=env_name] [--editor-cmd=...] <name>
```
Generates migration template

```
$ symfony doctrine:generate-migrations-db [--application=app_name] [--env=env_name]
```
Generates migration classes from existing database connections

```
$ symfony doctrine:generate-migrations-diff [--application=app_name] [--env=env_name]
```
Generate migration classes by producing a diff between your old and new schema

```
$ symfony doctrine:generate-migrations-models [--application=app_name] [--env=env_name]
```
Generates migration classes from an existing set of models

```
$ symfony doctrine:generate-module [--theme=theme_name] [--generate-in-cache] [--non-verbose-templates] [--with-show]
[--singular=singular_name] [--plural=plural_name] [--route-prefix=route_prefix] [--with-doctrine-route] [--env=env_name]
[--actions-base-class=base_class_for_actions] <app_name> <module_name> <model_class_name>
```
Generates a Doctrine module

```
$ symfony doctrine:generate-module-for-route [--theme=them_name] [--non-verbose-templates] [--singular=singular_name]
[--plural=plural_name] [--env=env_name] [--actions-base-class=base_class_for_actions] <app_name> <route_name>
```
Generates a Doctrine module for a route definition

```
$ symfony doctrine:insert-sql [--application=app_name] [--env=env_name]
```
Inserts SQL for current model. The task connects to the database and creates tables for all the lib/model/doctrine/*.class.php files.

```
$ symfony doctrine:migrate [--application=app_name] [--env=env_name] [--up] [--down] [--dry-run] [version]
```
Migrates database to current/specified version.

CLI (Tasks) - Database - Propel ORM

PROPEL TASKS

$ symfony propel:build [--application=app_name] [--env=env_name] [--no-confirmation] [--all] [--all-classes] [--model] [--forms] [--filters] [--sql] [--db] [--and-load=fixture_file] [--and-append=fixture_file]

Generates code based on your schema. You must specify what you would like built. For instance, if you want model and form classes built use the --model and --forms options: $ symfony propel:build --model --forms

$ symfony propel:build-all [--application=app_name] [--env=env_name] [--connection=conn_name] [--no-confirmation] [-F|--skip-forms] [-C|--classes-only] [--phing-arg=arbitrary_phing_arguments]

Generates Propel model and form classes, SQL and initializes the database

$ symfony propel:build-all-load [--application=app_name] [--env=env_name] [--connection=conn_name] [--no-confirmation] [-F|--skip-forms] [-C|--classes-only] [--phing-arg=arbitrary_phing_arguments] [--append] [--dir=fixture_dir]

Generates Propel model and form classes, SQL, initializes the database, and loads data

$ symfony propel:build-filters [--connection=conn_name] [--model-dir-name=model_dir_name] [--filter-dir-name=filter_form_dir_name] [--application=app_name] [--generator-class=generator_class]

Creates filter form classes from the schema for the current model. Read the schema information in config/*schema.xml and/or config/*schema.yml from the project and all installed plugins. The task use the propel connection as defined in config/databases.yml. The model filter form classes files are created in lib/filter. This task never overrides custom classes in lib/filter. It only replaces base classes generated in lib/filter/base

$ symfony propel:build-forms [--connection=conn_name] [--model-dir-name=model_dir_name] [--form-dir-name=form_dir_name] [--application=app_name] [--generator-class=generator_class_name]

Creates form classes for the current model. Read the schema information in config/*schema.xml and/or config/*schema.yml from the project and all installed plugins. The model form classes files are created in lib/form. This task never overrides custom classes in lib/form. It only replaces base classes generated in lib/form/base
E.g.: $ symfony propel:build-forms --connection="name"

$ symfony propel:build-model [--phing-arg=arbitrary_phing_arguments]

Creates model classes from the schema. The model classes files are created in lib/model. This task never overrides custom classes in lib/model. It only replaces files in lib/model/om and lib/model/map.

$ symfony propel:build-schema [--application=app_name] [--env=env_name] [--connection=conn_name] [--xml] [--phing-arg=arbitrary_phing_arguments]

Creates a schema from an existing database

$ symfony propel:build-sql [--phing-arg=arbitrary_phing_arguments]

Creates SQL statements for table creation. The generated SQL is optimized for the database configured in config/propel.ini

$ symfony propel:data-dump [--application=app_name] [--env=env_name] [--connection=conn_name] [--classes=...] [target]

Dumps database data to the fixtures directory. E.g.: $ symfony propel:data-dump > data/fixtures/dump.yml.
The task will dump data in data/fixtures/%target%.
E.g.: $ symfony propel:data-dump --classes="Article,Category"

$ symfony propel:data-load [--application=app_name] [--env=env_name] [--append] [--connection=conn_name] [dir_or_file1] ... [dir_or_fileN]

Loads data fixtures into the database. Loads data from all the files found in data/fixtures/
E.g.: $ symfony propel:data-load --application=frontend
E.g.2: $ symfony propel:data-load data/fixtures/dev data/fixtures/users.yml

$ symfony propel:generate-admin [--module=module_name] [--theme=theme_name] [--singular=singular_name] [--plural=plural_name] [--env=env_name] [--actions-base-class=base_class_for_actions] <app_name> <route_or_model>

Generates a Propel admin module.
E.g.: $ symfony propel:generate-admin frontend Article

$ symfony propel:generate-module [--theme=theme_name] [--generate-in-cache] [--non-verbose-templates] [--with-show] [--singular=singular_name] [--plural=plural_name] [--route-prefix=route_prefix] [--with-propel-route] [--env=env_name] [--actions-base-class=base_class_for_actions] <app_name> <module_name> <model_name>

Generates a Propel module. You can also create an empty module that inherits its actions and templates from a runtime generated module in %sf_app_cache_dir%/modules/auto%module% by using the --generate-in-cache option:
$ symfony propel:generate-module --generate-in-cache frontend article Article
You can change the default actions base class (default to sfActions) of the generated modules:
E.g.: $ symfony propel:generate-module --actions-base-class="ProjectActions" frontend article Article

$ symfony propel:generate-module-for-route [--theme=them_name] [--non-verbose-templates] [--singular=singular_name] [--plural=plural_name] [--env=env_name] [--actions-base-class=base_class_for_actions] <app_name> <route_name>

Generates a Propel module for a route definition

$ symfony propel:graphviz [--phing-arg=arbitrary_phing_arguments]

Creates a graphviz DOT visualization for automatic graph drawing of object model

$ symfony propel:insert-sql [--application=app_name] [--env=env_name] [--connection=conn_name] [--no-confirmation] [--phing-arg=arbitrary_phing_arguments]

Inserts SQL for current model. The task connects to the database and executes all SQL statements found in config/sql/*schema.sql files

$ symfony propel:schema-to-xml

Creates schema.xml from schema.yml

$ symfony propel:schema-to-yml

Converts XML schemas to YML

CLI (Tasks)

PLUGIN TASKS

```
$ symfony plugin:add-channel <name>
```
Adds a new PEAR channel
E.g.: symfony plugin:add-channel symfony.plugins.pear.example.com

```
$ symfony plugin:install
[-s|--stability=[stable|beta|alpha]]
[-r|--release=preferred_version]
[-c|--channel=PEAR_channel_name]
[-d|--install_deps] [--force-license] <name>
```
Installs a plugin. By default, it installs the latest stable release.
To force installation of all required dependencies, use the install_deps flag

```
$ symfony plugin:list
```
Lists all installed plugins.
Also gives the channel and version for each plugin

```
$ symfony plugin:publish-assets [--core-only]
[plugins1] ... [pluginsN]
```
Publishes web assets for all plugins

```
$ symfony plugin:uninstall
[-c|--channel=PEAR_channel_name]
[-d|--install_deps] <name>
```
Uninstalls a plugin

```
$ symfony plugin:upgrade
[-s|--stability=[stable|beta|alpha]]
[-r|--release=preferred version]
[-c|--channel=PEAR_channel_name] <name>
```
Upgrades a plugin. The default channel is symfony.
If the plugin contains some web content (images, css or js),
the task also updates the web/%name% directory content on Windows.

TEST TASKS

```
$ symfony symfony:test [-u|--update-autoloader]
[-f|--only-failed][--xml=file_name] [--rebuild-all]
```
Launches the symfony test suite

```
$ symfony test:all [-f|--only-failed] [--xml=file_name]
```
Launches all unit and functional tests found in test/. If some tests fail, you can use the --trace option to have more information about the failures: $ symfony test:all -t

```
$ symfony test:coverage [--detailed] <test_name> <lib_name>
```
Outputs the test code coverage given a test file or test directory and a lib file or lib directory for which you want code coverage
E.g.: $ symfony test:coverage test/unit/model lib/model

```
$ symfony test:functional [--xml="..."] <app_name>
[controller1] ... [controllerN]
```
Launches functional tests for a given application. The task launches all tests found in test/functional/%application%

```
$ symfony test:unit [--xml=filename] [name1]...[nameN]
```
Launches all unit tests found in test/unit.

I18N TASKS

```
$ symfony i18n:extract [--display-new] [--display-old]
[--auto-save] [--auto-delete] <app_name> <culture>
```
Extracts i18n strings from your project files for the given application and target culture. E.g.: $ symfony i18n:extract frontend fr

```
$ symfony i18n:find [--env=env_name] <app_name>
```
Finds non internationalized strings embedded in templates. Is able to find non internationalized strings in pure HTML and in PHP code

LOG TASKS

```
$ symfony log:clear
```
Clears all symfony log files

```
$ symfony log:rotate [--history=max_number_old_log_files_to_keep]
[--period=period_in_days] <app_name> <env_name>
```
Rotates an application's log files

TASK PACKAGE

sfAppRoutesTask
sfBaseTask
sfCacheClearTask
sfCommandApplicationTask
sfConfigureAuthorTask
sfDeprecatedClassesValidation
sfDeprecatedConfigurationFilesValidation
sfDeprecatedHelpersValidation
sfDeprecatedMethodsValidation
sfDeprecatedPluginsValidation
sfDeprecatedSettingsValidation
sfDoctrineBuildTask
sfDoctrineConfigureDatabaseTask
sfGenerateAppTask
sfGenerateModuleTask
sfGenerateProjectTask
sfGenerateTaskTask
sfGeneratorBaseTask
sfHelpTask
sfI18nExtractTask
sfI18nFindTask
sfListTask
sfLogClearTask
sfLogRotateTask
sfParameerHolderValidation
sfPluginAddChannelTask
sfPluginBaseTask
sfPluginInstallTask
sfPluginListTask
sfPluginPublishAssetsTask
sfPluginUninstallTask
sfPluginUpgradeTask
sfProjectClearControllersTask
sfProjectDeployTask
sfProjectDisableTask
sfProjectEnableTask
sfProjectOptimizeTask
sfProjectPermissionsTask
sfProjectSendEmailsTask
sfPropelBuildTask
sfPropelConfigureDatabaseTask
sfSymfonyTestTask
sfTask
sfValidateTask
sfValidation

You can pass an associative array of arguments and options to sfTask::run():

```
$task = new sfDoctrineConfigureDatabaseTask(
$this->dispatcher,$this->formatter);
$task->run(
       array('dsn'  => 'mysql:dbname=mydb;host=localhost'),
       array('name' => 'master')
      );
```

sfTask Class

addArgument()
addArguments($arguments)
addOption()
addOptions($options)
ask($question, $style, $default,)
askAndValidate($question, $validator, $options)
askConfirmation($question, $style, $default,)
asXml()
configure()
doRun()
execute($arguments, $options)
getAliases()
getArguments()
getBriefDescription()
getDetailedDescription()
getFormatter()
getFullName()
getName()
getNamespace()
getOptions()
getSynopsis()
initialize($dispatcher, $formatter)
log($messages)
logBlock($messages, $style)
Logs a message as a block of text.
logSection($section, $message, $size, $style)
process()
run($arguments, $options)
runFromCLI($commandManager, $options)
setFormatter()
strlen()

E.g.:

```
$answer = $this->askAndValidate(
                'What is you email?',
                new sfValidatorEmail());
```