Struttura delle cartelle e CLI (Tasks)

STRUTTURA DELLE CARTELLE

```
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
```

INTERFACCIA A LINEA DI COMANDO (CLI) - TASKS

Uso: `symfony [opzioni] task_name [argomenti]`

Opzioni e argomenti di default per ogni task di symfony:

`--help`	`-H`	mostra questo messaggio di aiuto
`--quiet`	`-q`	Non invia messaggi allo standard output
`--trace`	`-t`	Abilita la tracciatura invoke/execute tracing, abilita il backtrace completo
`--version`	`-V`	Mostra la versione del programma
`--color`		Forza l'output in colori ANSI
`--xml`		Output dell'help in formato XML

Il nome di un **task** può essere costituito da un namespace opzionare (ad es. generate, project, ...) e da un nome, separati dai duepunti (:)

```
$ symfony
```
Lista completa dei task disponibili

```
$ symfony list [--xml] [namespace]
```
Lista dei task

```
$ symfony -V
```
Versione installata del package di symfony e percorso delle librerie di symfony usate dal CLI

```
$ symfony help [--xml] [task_name]
```
Mostra l'help per un task

```
$ symfony cache:clear [--app=app_name] [--env=[prod|dev]] [--type=type]
```
Cancella le informazioni memorizzate nella cache (shortcut: cc)
I tipi predefiniti sono: config, i18n, routing, module e template

```
$ symfony configure:author <author_name>
```
Configura l'autore per un progetto. Usato dal generatore per preconfigurare l'header dei PHPDoc

```
$ symfony configure:database [--env=env_name] [--name=conn_name]
  [--class=db_class_name] [--app=app_name] <dsn> [username] [password]
```
Configura il database DSN per un progetto

```
$ symfony app:routes <app_name> [route_name]
```
Mostra le routes correnti per un'applicazione

generate

```
$ symfony generate:task [--dir=[lib/task]] [--use-database=
  [doctrine|propel|main|false]] [--brief-description="..."] <task_name>
```
Crea una classe scheletro per un nuovo task

```
$ symfony generate:project [--orm=[Doctrine|Propel|none]]
  [--installer=installer_script] <project_name> [author_name]
```
Genera un nuovo progetto. Crea una struttura di cartelle di base per un nuovo progetto.

```
$ symfony generate:app [--escaping-strategy=...] [--csrf-secret=...] <app_name>
```
Genera una nuova applicazione. Crea una struttura di cartelle di base per una nuova applicazione nel progetto corrente. Inoltre crea due script front controller nella cartella web/: <app_name>.php per l'ambiente di produzione and <app_name>_dev.php per l'ambiente di sviluppo.

```
$ symfony generate:module <app_name> <module_name>
```
Genera un nuovo modulo. Crea la struttura di cartelle di base per un nuovo modulo in un'applicazione esistente.

project

```
$ symfony project:clear-controllers
```
Cancella tutti i controller che non siano dell'ambiente di produzione.

```
$ symfony project:deploy [--go] [--rsync-dir=config]
[--rsync-options=[-azC|--force|--delete|--progress]] <server_name>
```
Distribuisce un progetto a un server. Il server dev'essere configurato in config/properties.ini --rsync-dir è la cartella contenente i files rsync*.txt

```
$ symfony project:disable <env_name> [app1] ... [appN]
```
Disabilita un'applicazione in un dato ambiente

```
$ symfony project:enable <env_name> [app1] ... [appN]
```
Abilita un'applicazione in un dato ambiente

```
$ symfony project:optimize <app_name> [env_name]
```
Ottimizza un progetto per le migliori prestazioni. Questo task dovrebbe essere usato solo in un server di produzione

```
$ symfony project:permissions
```
Corregge i permessi della cartella

```
$ symfony project:send-emails [--application=app_name] [--env=env_name]
[--message-limit=max_number_msg_to_send] [--time-limit=time_limit_in_sec]
```
Invia le email memorizzate in una coda

```
$ symfony project:validate
```
Rileva utilizzi deprecati nel tuo progetto

CLI (Tasks) - Database - ORM Doctrine

TASK DI DOCTRINE

```
$ 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]
```
Genera codice basato sul tuo schema

```
$ symfony doctrine:build-db [--application=app_name] [--env=env_name] [database1] ... [databaseN]
```
Crea uno o più database basati sulla configurazione in config/database.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]
```
Crea classi di filtro per i form dallo schema. Le classi sono create in lib/doctrine/filter. Non sovrascrive in nessun caso le classi personalizzate in lib/doctrine/filter. Rimpiazza unicamente le classi di base generate 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]
```
Crea classi per i form dallo schema per il modello corrente. Le classi sono create in lib/doctrine/form. Questo task non sovrascrive le classi personalizzate in lib/doctrine/form, ma rimpiazza unicamente le classi di base generate in lib/doctrine/form/base

```
$ symfony doctrine:build-model [--application=app_name] [--env=env_name]
```
Crea classi modello dallo schema. Legge le informazioni dello schema contenute in config/doctrine/*.yml dal progetto e da tutti i plugin abilitati.
I file delle classi modello sono creati in lib/model/doctrine.
Questo task non sovrascrive le classi personalizzate in lib/model/doctrine, ma rimpiazza unicamente i file in lib/model/doctrine/base

```
$ symfony doctrine:build-schema [--application=app_name] [--env=env_name]
```
Crea uno schema da un database esistente. Il task crea un file yml in config/doctrine

```
$ symfony doctrine:build-sql [--application=app_name] [--env=env_name]
```
Crea SQL per il modello corrente. L'SQL generato è ottimizzato per il database configurato in config/databases.yml

```
$ symfony doctrine:clean-model-files [--no-confirmation]
```
Cancella tutte le classi modello generate per modelli che non sono più presenti nel tuo schema YAML. Alias: doctrine:clean

```
$ symfony doctrine:create-model-tables [--application=app_name] [--env=env_name] [models1] ... [modelsN]
```
Cancella e ricrea le tabelle per i modelli specificati

```
$ symfony doctrine:data-dump [--application=app_name] [--env=env_name] [target_filename]
```
Esegue un dump del database in data/fixtures/%target_filename%. Il file di dump è in formato YML e può essere reimportato usando il task doctrine:dataload

```
$ symfony doctrine:data-load [--application=app_name] [--env=env_name] [--append] [dir_or_file1] ... [dir_or_fileN]
```
Carica dati da tutti i files trovati in data/fixtures. Usando l'opzione --append, il task non cancella i dati correnti del database

```
$ symfony doctrine:delete-model-files [--no-confirmation] model_name1 ... [model_nameN]
```
Cancella tutti i files associati con certi modelli

```
$ symfony doctrine:dql [--application=app_name] [--env=env_name] [--show-sql] [--table] <dql_query> [param1] ... [paramN]
```
Esegue una query DQL e mostra i risultati formattati.
Ad es. $ 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]
```
Cancella il database per il modello corrente

```
$ 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>
```
Genera un modulo amministrativo di Doctrine

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

```
$ symfony doctrine:generate-migrations-db [--application=app_name] [--env=env_name]
```
Genera delle classi di migrazione dalle connessioni al database esistenti

```
$ symfony doctrine:generate-migrations-diff [--application=app_name] [--env=env_name]
```
Genera delle classi di migrazione producendo un diff tra il vecchio e il nuovo schema

```
$ symfony doctrine:generate-migrations-models [--application=app_name] [--env=env_name]
```
Genera delle classi di migrazione da un insieme esistente di modelli

```
$ 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>
```
Genera un modulo di Doctrine

```
$ 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>
```
Genera un modulo di Doctrine per la definizione di una route

```
$ symfony doctrine:insert-sql [--application=app_name] [--env=env_name]
```
Inserisce l'SQL per il modello corrente. Il task si connette al database e crea le tabelle per tutti i file in lib/model/doctrine/*.class.php

```
$ symfony doctrine:migrate [--application=app_name] [--env=env_name] [--up] [--down] [--dry-run] [version]
```
Migra il database alla versione corrente o specificata

CLI (Tasks) - Database - Propel ORM

TASK DI PROPEL

```
$ 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]
```

Genera codice basato sul tuo schema. Devi specificare cosa desideri costruire. Ad esempio se desideri la costruzione delle classi per il modello e i form usa le opzioni --model e --forms: $ 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]
```

Genera il modello Propel e le classi per i form, l'SQL e inizializza il 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]
```

Genera il modello Propel e le classi per i form, l'SQL, inizializza il database e carica i dati

```
$ 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]
```

Crea le classi per i filtri form dallo schema del modello corrente. Legge le informazioni dello schema in config/*schema.xml e/o config/*schema.yml del progetto corrente e di tutti i plugin installati. Il task usa la connessione propel come definito in config/databases.yml. I file delle classi dei filtri form sono creati in lib/filter. Questo task in nessun caso sovrascrive le classi personalizzate in lib/filter, ma rimpiazza unicamente le classi di base generate 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]
```

Crea le classi form per il modello corrente. Legge le informazioni dello schema in config/*schema.xml e/o config/*schema.yml del progetto e di tutti i plugin installati. I file delle classi dei form sono creati in lib/form. Questo task in nessun caso sovrascrive le classi personalizzate in lib/form, ma rimpiazza unicamente le classi di base generate in lib/form/base. Ad esempio $ symfony propel:build-forms --connection="name"

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

Crea le classi del modello dallo schema. I file delle classi del modello sono creati in lib/model. Questo task in nessun caso sovrascrive le classi personalizzate in lib/model, ma rimpiazza unicamente i file in lib/model/om e lib/model/map

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

Crea uno schema da un database esistente

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

Crea delle dichiarazioni SQL per la creazione della tabella. L'SQL generato è ottimizzato per il database configurato in config/propel.ini

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

Esegue un dump del database nella cartella delle fixtures. Ad es. $ symfony propel:data-dump > data/fixtures/dump.yml. Il task esegue il dump dei dati in data/fixtures/%target%.
Ad es. $ 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]
```

Carica le fixtures nel database. Carica i dati da tutti i file trovati in data/fixtures/
Es.: $ symfony propel:data-load --application=frontend
Es.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>
```

Genera un modulo amministrativo di Propel.
Ad es.: $ 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>
```

Genera un modulo di Propel. Puoi anche creare un modulo vuoto che erediti le sue azioni e template da un modulo generato in runtime in %sf_app_cache_dir%/modules/auto%module% usando l'opzione --generate-in-cache:
$ symfony propel:generate-module --generate-in-cache frontend article Article
Puoi cambiare la classe di base delle azioni di default (default to sfActions) dei moduli generati:
Es.: $ 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>
```

Genera un modulo Propel per la definizione di una route

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

Crea una visualizzazione graphviz DOT per il tracciamento automatico del grafico del modello a oggetti

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

Inserisce l'SQL per il modello corrente. Il task si connette al database ed esegue tutte le dichiarazioni SQL trovate nei file in config/sql/*schema.sql

```
$ symfony propel:schema-to-xml
```

Crea il file schema.xml partendo da schema.yml

```
$ symfony propel:schema-to-yml
```

Converte lo schema XML in YML

CLI (Tasks)

TASK DEI PLUGIN

```
$ symfony plugin:add-channel <name>
```
Aggiunge un nuovo canale PEAR
Es.: 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>
```
Installa una plugin. Per default install l'ultima release stabile.
Per forzare l'installazione di tutte le dipendenza usare il floag install_deps

```
$ symfony plugin:list
```
Elenca tutte le plugin installate.
Inoltre visualizza il canale e la versione per ogni plugin

```
$ symfony plugin:publish-assets [--core-only]
[plugins1] ... [pluginsN]
```
Pubblica l'assets per tutte le plugin

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

```
$ symfony plugin:upgrade
[-s|--stability=[stable|beta|alpha]]
[-r|--release=preferred version]
[-c|--channel=PEAR_channel_name] <name>
```
Aggiorna una plugin. Il canale di default è symfony.
Se una plugin contiene contenuti web (immagini, css o js) il task inoltre aggiorna la cartella web/%name%

TASK DEI TEST

```
$ symfony symfony:test [-u|--update-autoloader]
[-f|--only-failed][--xml=file_name] [--rebuild-all]
```
Avvia la suite symfony per i test

```
$ symfony test:all [-f|--only-failed] [--xml=file_name]
```
Esegue tutti i test unitari e funzionali che trova in test/. Se qualche test fallisce si può usare l'opzione --trace per avere maggiori informazioni sul fallimento: $ symfony test:all -t

```
$ symfony test:coverage [--detailed] <test_name> <lib_name>
```
Visualizza la copertura del codice di un dato file di test o di una directory di test edi una lib o directory lib.
Es.: $ symfony test:coverage test/unit/model lib/model

```
$ symfony test:functional [--xml="..."] <app_name>
[controller1] ... [controllerN]
```
Esegue i test funzionali per una specifica applicazione. Il task esegue e test trovati in test/functional/%application%

```
$ symfony test:unit [--xml=filename] [name1]...[nameN]
```
Esegue tutti i test unitari trovati in test/unit

TASK DI I18N

```
$ symfony i18n:extract [--display-new] [--display-old]
[--auto-save] [--auto-delete] <app_name> <culture>
```
Estrae le stringhe i18n dai file del progetto di una specifica applicazione e una specifica cultura.
Es.: $ symfony i18n:extract frontend it

```
$ symfony i18n:find [--env=env_name] <app_name>
```
Trova stringhe non internazionalizzate incluse nei template. E' in grado di trovare stringhe non internazionalizzate anche in puro HTML o codice PHP

TASK DI LOG

```
$ symfony log:clear
```
Cancella tutti i file di log di symfony

```
$ symfony log:rotate [--history=max_number_old_log_files_to_keep]
[--period=period_in_days] <app_name> <env_name>
```
Rotazione dei file di log di una applicazione

TASK DI 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

Puoi passare un array associativo di argomenti e opzioni a `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()

Es.:

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