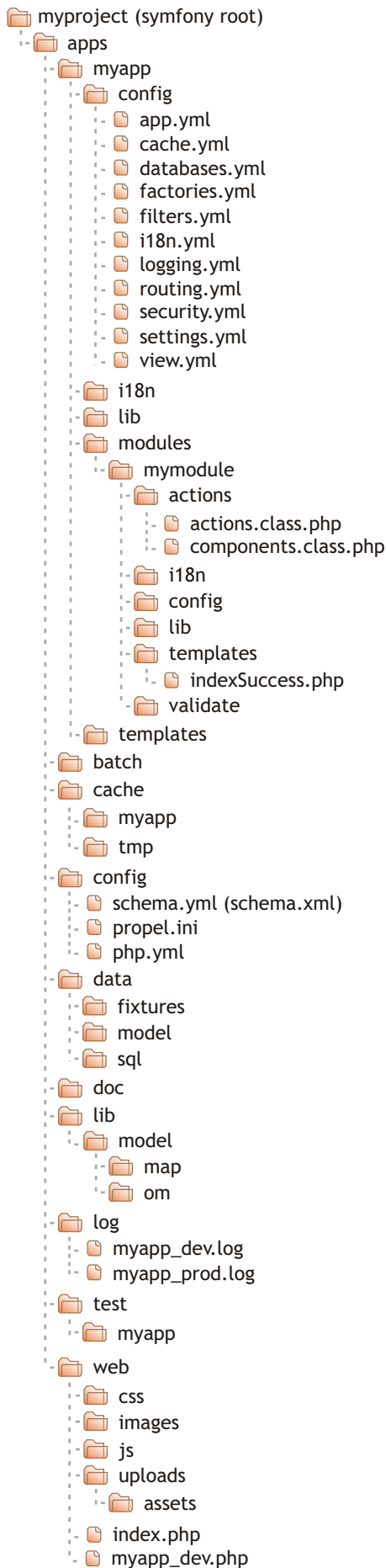


Verzeichnisstruktur und CLI

Vorgegebene Verzeichnisstruktur



Kommandozeilen-Interface (CLI)

```

$ symfony -T
Gibt eine Liste aller verfügbaren Administrationsbefehle aus

$ symfony -V
Gibt die installierte symfony Version aus

$ symfony clear-cache <application_name> [template|config]
Löscht den Cache (kurz: cc)

$ symfony init-project <project_name>
Initialisiert ein Projekt und generiert die, für die Laufzeitumgebung, benötigten Ordner und Dateien

$ symfony init-app <application_name>
Initialisiert eine Anwendung. Zusätzlich werden im Verzeichnis web/ folgende PHP-Dateien für die Front Controller der Standardumgebung erstellt:
index.php (produktion) und myapp_dev.php (entwicklung)

$ symfony init-module <application_name> <module_name>
Initialisiert ein neues Modul. Anschließend kann das Modul verwendet unter folgenden Adresse werden: http://myapp.example.com/index.php/mymodule

$ symfony propel-build-schema [xml]
Generiert die schema.yml aus einer bestehenden Datenbank heraus.
Um eine schema.xml zu generieren, kann der xml Parameter verwendet werden.

$ symfony propel-build-model
Generiert, entsprechend dem Datenmodell der schema.yml, die PHP-Klassen für ein Modell. Die grundlegenden Zugriffsklassen werden automatisch im Verzeichnis myproject/lib/model/om/ generiert:
BaseArticle.php      BaseComment.php
BaseArticlePeer.php  BaseCommentPeer.php
Zusätzlich werden die eigentlichen Zugriffsklassen im Verzeichnis myproject/lib/model erstellt:
Article.php          Comment.php
ArticlePeer.php      CommentPeer.php

$ symfony propel-generate-crud <application_name> <module_name> <ClassName>
Generiert, basierend auf der Klasse eines Modells, ein neues Propel CRUD Modul

$ symfony propel-build-sql
Erstellt den SQL-Code für die Tabellen der schema.yml in der Datei
myproject/data/sql/lib.model.schema.sql

$ symfony propel-build-db
Erstellt eine leere Datenbank

$ symfony propel-insert-sql
Fügt den SQL-Code aus der Datei myproject/data/sql/lib.model.schema.sql in die Datenbank ein

$ symfony sync <environment_name> [go]
Synchronisiert das Projektverzeichnis mit einer anderen Maschine

$ symfony propel-init-admin <application_name> <module_name> <ClassName>
Initialisiert, basierend auf der Klasse eines Modells, ein neues Propel Admin Modul

$ symfony test <application_name>
Startet die Testsuite für eine Anwendung

$ symfony plugin-install [local|global] <channel_name>/<plugin_name>
Installiert ein neues Plugin

$ symfony freeze
Konvertiert das Projekt zu einer unabhängigen und alleinstehenden Anwendung

$ symfony unfreeze
Bringt ein Projekt in den ursprünglichen Zustand zurück. Die Verzeichnisse data/symfony/, lib/symfony/ und web/sf/ werden gelöscht.

$ symfony disable <application_name> <environment_name>
Temporäres deaktivieren der Anwendung, um z.B. Bibliotheken oder große Datensätze zu aktualisieren

$ symfony enable <application_name> <environment_name>
Reaktiviert die Anwendung und löscht den cache

$ symfony clear-controllers
Löscht alle Controller im web/ Verzeichnis die nicht teil einer Produktionsumgebung sind. Falls die Front Controller der Entwicklungsumgebung nicht in der rsync_exclude.txt stehen, gibt dieser Befehl einem die Sicherheit das keine Hintertüren die Interna der Anwendung preisgeben.

$ symfony fix-perms
Behebt Probleme mit Verzeichnisrechten indem die Rechte von log/ und cache/ auf 0777 gesetzt werden (Beide Verzeichnisse müssen für die korrekte funktionsweise des Frameworks beschreibbar sein)

$ symfony log-purge
Löscht die symfony Log-Dateien in Anwendungen sowie Umgebungen die purge: on (welches die Voreinstellung ist) in der logging.yml Datei stehen haben
    
```