O'REILLY®

TYPO3

Theming und Distribution

DEN NEUEN STANDARD EFFEKTIV EINSETZEN

Thomas Deuling,
Jo Hasenau, Kay Strobach

Inhalt

Teil II TYPO3-Themes erstellen

KAPITEL 7

Theme-Struktur anlegen

Bevor Sie nun damit starten, Ihr erstes eigenes Theme zu erstellen, werden wir Ihnen zeigen, wie Themes struktiert sein sollten. Dies bezieht sich zum einen auf die Vererbungshierarchie, auf die wir zuerst eingehen werden, aber auch auf die Verzeichnis- und Datenstruktur des eigentlichen Themes. Bei der Entwicklung von Themes setzen wir auf sogenannte Vererbung. Das heißt, dass bestimmte Funktionalitäten aus dem eigentlichen Theme in eigenständige Erweiterungen ausgelagert sind und wir dann später auf diese aufsetzen. Das hat den Vorteil, dass alle Themes, die auf diesen Erweiterungen basieren, strukturell bei den wichtigsten Funktionen immer identisch sind. Dies wiederum erleichtert Ihnen, später einen Theme-Wechsel vorzunehmen, ohne dass Ihnen Daten oder Einstellungen verloren gehen. Der Vererbungsbaum in Abbildung 7-1 soll das veranschaulichen.

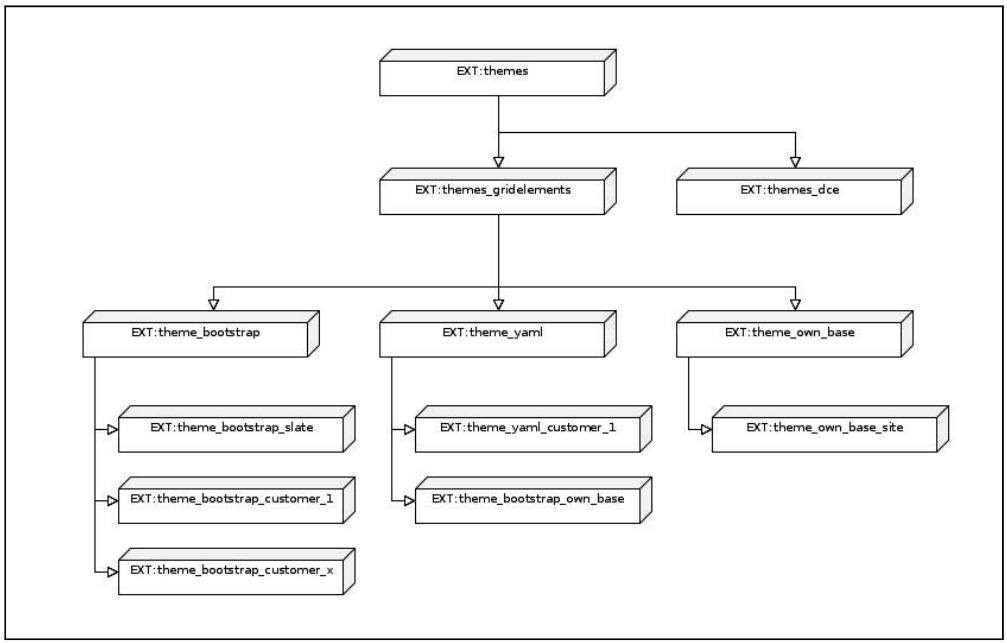

Abbildung 7-1: Theme-Vererbungsbaum

EXT:themes
: Auf der obersten Ebene befindet sich die Erweiterung THEMES, die die Basis für alle Themes bereitstellt. Diese Basis erhalten Sie über das statische Template der THEMES-Erweiterung, und sie wird beim späteren Auswählen eines konkreten Themes automatisch ins System injiziert.

*EXT:themes_**
: Auf der nächsten Ebene finden Sie die Erweiterung *themes_gridelements*, die grundlegende Strukturen und Bibliotheken des Themes bereitstellt. Diese basieren, wie der Name schon sagt, auf *Gridelements* und stellen Ihnen beispielsweise Backend-Layouts und Grid-Elemente zur Verfügung. Auf dieser Ebene könnte es später, nach derselben Logik wie in *themes_gridelements* gezeigt, auch eine Erweiterung *themes_dce* geben, die die gleichen Bibliotheken bereitstellt, die lediglich durch geringfügige Modifikationen auf die *dce*-Erweiterung zugeschnitten sind.

*EXT:theme_**
: Auf diesen beiden Erweiterungen aufbauend, entsteht das eigentliche Theme. Es bekommt durch die beiden darunterliegenden Erweiterungen schon eine Vielzahl an Funktionen bereitgestellt. Hierin implementieren Sie nun das konkrete Aussehen Ihres Themes.

Aber das ist noch nicht alles. Angenommen, Sie hätten nun ein *theme_bootstrap* implementiert und es wäre schon in einigen TYPO3-Instanzen im Einsatz, Sie bräuchten jedoch in einer bestimmten Instanz eine neue Funktion oder ein abweichendes Aussehen. Dann wären Sie durch die standardisierte Struktur in der Lage, noch einmal auf eben diese Theme-Erweiterung aufzusetzen, ihr komplettes Aussehen und Verhalten zu übernehmen, aber die relevanten Teile anzupassen.

Wie diese Vererbungsstruktur im Detail funktioniert, werden Sie im Laufe des Buchs kennenlernen. Wir wollen Ihnen in erster Linie zeigen, wie Sie ein Theme wie unter *theme_bootstrap* gezeigt erstellen oder später leicht auf fertige Themes aufsetzen können.

Bevor Sie nun wirklich loslegen können, sollten Sie sich kurz Gedanken darüber machen, wie ein solches Theme am besten strukturiert werden sollte und wo es innerhalb von TYPO3 abgelegt wird. Das Vorgehen bei der Theme-Entwicklung weicht von Entwickler zu Entwickler sehr voneinander ab, und es gibt eine Vielzahl an Möglichkeiten. Zuallererst sollten Sie Ihrem Theme aber einen Namen geben, wir haben uns bei unserem Theme für den Namen *Theme-Bootstrap* (*theme_bootstrap*) entschieden.

Sie können sich das Theme, das wir in diesem Buch vorstellen, unter folgender Adresse herunterladen:Theme-Bootstrap Download (*https://github.com/typo3-themes/theme_bootstrap*).

In diesem ersten Schritt geht es nun darum, den besten Ablageort und die optimale Verzeichnisstruktur für das Theme zu finden.

Speicherplatz eines Themes – Wo sollten die Dateien des Themes liegen?

Eine Möglichkeit wäre, das komplette Theme im *fileadmin*-Verzeichnis der TYPO3-Installation abzulegen. Die Erfahrung hat jedoch gezeigt, dass dies nicht der beste Platz für Dateien dieser Art ist. Wenn beispielsweise einem Redakteur Zugriff auf das komplette *fileadmin*-Verzeichnis gewährt wird, hat dieser auch Zugriff auf die Dateien des Themes und könnte somit, je nach Konfiguration der TYPO3-Installation, auch die Dateien des Themes verändern. Dies sollte unbedingt verhindert werden. Der Redakteur sollte immer nur die Möglichkeit haben, die Inhalte selbst zu bearbeiten.

Ein anderer, besserer Speicherort für ein Theme ist die Bereitstellung des Themes in einer TYPO3-Erweiterungsstruktur. Dies hat viele Vorteile. So kann das Theme beispielsweise über den Erweiterungsmanager importiert und auch exportiert werden. Man hat eine saubere Trennung der Theme-Dateien von den Content-Dateien, und falls es doch erforderlich sein sollte, dem Administrator der Seite das Bearbeiten von Theme-Dateien über das Modul *Dateiliste* zu erlauben, kann für das entsprechende Verzeichnis einfach ein TYPO3-Dateispeicher angelegt werden. Auf diesen Dateispeicher sollten die Redakteure dann natürlich ebenfalls keinen Zugriff bekommen. Der wichtigste Aspekt für diese Entscheidung ist, dass Sie dadurch eine Anbindung an die THEMES-Erweiterung bereitstellen können, sodass das Theme später über den Theme-Selektor auswählbar und konfigurierbar ist.

Resultierend aus der Entscheidung, das Theme in einer TYPO3-Erweiterung einzubetten, erstellen Sie nun ein Verzeichnis namens *theme_bootstrap* im Verzeichnis */typo3conf/ext/* – in diesem werden Sie im Laufe des Buchs das neue Theme entwickeln.

Der Verzeichnisname eines Themes bzw. der Theme-Erweiterung ist gleichzeitig auch der sogenannte *Erweiterungsschlüssel* (*Extension-Key*) der Erweiterung. Bei der eigentlichen Benennung Ihres Themes sind Sie völlig frei, was beim Erweiterungsschlüssel nicht der Fall ist. Dieser muss nach den folgenden Regeln aufgebaut sein: Es sind nur die Buchstaben a bis z (kleingeschrieben), die Zahlen 0 bis 9 und ein _ (Unterstrich) als enthaltene Zeichen erlaubt. Er darf nicht mit den folgenden Präfixen beginnen: `tx_`, `user_`, `pages_`, `tt_`, `sys_`, `ts_language_`, `csh_`. Er muss mit einem Buchstaben von a bis z beginnen. Er muss mindestens drei Zeichen und darf maximal 30 Zeichen lang sein (Unterstriche nicht mitgerechnet).

Verzeichnisstruktur eines Themes

Da nun geklärt ist, an welchem Ort das Theme ablegt wird, können Sie sich dem nächsten Punkt widmen: Welche Verzeichnisstruktur sollte ein Theme haben, wenn Sie es in einer Erweiterung eingebettet verwenden wollen?

Dabei sollte der Fokus nicht nur auf einer reinen Erweiterung für TYPO3 CMS liegen, es sollte auch die Struktur eines TYPO3-Neos-Site-Package berücksichtigt werden, sodass eine eventuelle spätere Portierung Ihres TYPO3-Themes von TYPO3 CMS zu TYPO3 Neos nicht erschwert wird. Nachfolgend geben wir Ihnen schon mal eine Übersicht der möglichen Verzeichnisstruktur (siehe Beispiel 7-1), die Sie dann im Laufe dieses Kapitels erläutert bekommen.

Beispiel 7-1: Übersicht der Theme-Verzeichnisstruktur

```
typo3conf
- ext
  - theme_bootstrap
    - Configuration
      - BackendLayouts
      - Elements
      - PageTS
        - Library
        - tsconfig.txt
      - TypoScript
        - Library
        - constants.txt
        - setup.txt
      - UserTS
    - Documentation
    - Meta
      - Screenshots
        - screenshot.png
        - screenshot-01.png
        - screenshot-02.png
      - theme.yaml
    - Resources
      - Private
        - Dyncss
        - Extensions
          - News
            - Dyncss
            - Layouts
            - Partials
            - Templates
        - Language
        - Layouts
        - Partials
        - Templates
        - TypoScript
        - .htaccess
      - Public
        - Contrib
        - Extensions
          - News
            - Images
            - Stylesheets
        - Fonts
        - Icons
        - Images
        - JavaScript
```

```
        - Stylesheets
    - ext_emconf.php
    - ext_icon.gif
```

Beginnen Sie nun zunächst auf der obersten Ebene, auf der Sie die folgenden Hauptverzeichnisse erstellen müssen (siehe Beispiel 7-2):

Beispiel 7-2: Hauptverzeichnisse im Theme-Verzeichnis

```
typo3conf
- ext
  - theme_bootstrap
    - Configuration
    - Initialisation
    - Documentation
    - Meta
    - Resources
```

`Configuration`
: Hier liegt für gewöhnlich die Konfiguration einer TYPO3-Erweiterung. Sie übernehmen dieses Verzeichnis und legen auch hier später jegliche Dateien ab, die mit der Konfiguration zusammenhängen.

`Initialisation`
: Mithilfe dieses Verzeichnisses können Sie Ihre Erweiterung (bzw. unser Theme) ab TYPO3 6.2 mit Initialdaten ausstatten. Dies werden wir in Teil III, *TYPO3-Distributions*, genauer behandeln.

`Documentation`
: In diesem Verzeichnis kann eine Dokumentation zum jeweiligen Theme im ReST-Format abgelegt werden. Wie man eine solche Dokumentation erstellt, wird in diesem Buch nicht detailliert beschrieben, da es dafür hervorragende Dokumentationen auf *www.typo3.org* gibt.

`Meta`
: Hier werden Metadaten und Vorschaubilder zum Theme abgelegt.

`Resources`
: Hier legen Sie alle Ressourcen eines Themes ab. Dies können Fluid-Templates, JavaScript-Dateien, Stylesheets, Bilder und vieles mehr sein.

Weitere Informationen zum Erstellen von Dokumentationen für TYPO3-Erweiterungen finden Sie unter *http://docs.typo3.org/typo3cms/CoreApiReference/stable/ExtensionArchitecture/Documentation/* und unter http://wiki.typo3.org/ReST (*http://wiki.typo3.org/ReST*).

Das Verzeichnis Configuration

Das *Configuration*-Verzeichnis besteht mindestens aus einem `TypoScript`- und einem PageTS-Verzeichnis, in dem die benötigten TypoScript-Dateien abgelegt werden.

Das Verzeichnis *TypoScript* enthält in erster Linie die beiden Dateien *constants.txt* und *setup.txt*. Diese beiden Dateien enthalten, wie auch in üblichen TYPO3-Erweiterungen, das Basis-TypoScript für die Erweiterung bzw. für Ihr Theme. Unterhalb des *TypoScript*-Verzeichnisses gibt es noch ein Verzeichnis *Library*, in dem TypoScript-Dateien mit bestimmter Funktionalität wie beispielsweise Menüs, Marker oder ähnlichem abgelegt werden. Hier sollte der Dateiname möglichst gut den jeweiligen Inhalt der TypoScript-Datei widerspiegeln. Wichtig für die im Verzeichnis *Library* abgelegten Dateien ist die Dateiendung. Alle TypoScript-Setup -Dateien bekommen die Dateiendung *.setupts*, und alle TypoScript-Constants -Dateien bekommen die Dateiendung *.constantsts*. Dadurch ist es uns möglich, diese später sauber gefiltert auslesen zu können.

Des Weiteren gibt es im *Configuration*-Verzeichnis ein Verzeichnis namens *PageTS*, in dem eine Datei tsconfig.txt_ liegt, die das benötigte Seiten-TypoScript des Themes enthalten sollte. Seiten-TypoScript wird verwendet, um die Funktionalität im TYPO3-Backend zu konfigurieren. Auch hier gibt es wieder ein Verzeichnis *Library*, in dem kleinere Seiten-TypoScripte abgelegt werden können. Die Seiten-TypoScript-Dateien in diesem Verzeichnis sollten die Dateiendung *.pagets* bekommen.

Gemäß dem Motto *divide et impera* (teile und herrsche) versuchen wir hier, die TypoScripte nicht alle in eine große Datei zu zwängen, sondern diese in kleine logische Teile im *Library*-Verzeichnis abzulegen. Je besser diese nach Funktionalität und Aufgabe getrennt sind, desto leichter wird es Ihnen später fallen, kleine Funktionen schnell von einem Theme in das andere zu übernehmen.

Die Dateien *setup.txt*, *constants.txt* und *tsconfig.txt* werden beim Auswählen des Themes über die THEMES-Erweiterung automatisch in die Theme-Seite injiziert. Somit ist keine Zeile TypoScript im Root-Template oder in den Seiteneinstellungen notwendig. Das ist sehr wichtig, denn so haben Sie jegliches TypoScript in Ihre Theme-Erweiterung ausgelagert.

Im Folgenden sollten Sie sich zunächst alle bisher vorstellbaren Verzeichnisse im *Configuration*-Verzeichnis ansehen (siehe Beispiel 7-3):

Beispiel 7-3: Verzeichnisse im Configuration-Verzeichnis

```
typo3conf
- ext
  - theme_bootstrap
    - Configuration
      - BackendLayouts
      - Elements
      - PageTS
        - Library
      - TCA
      - TypoScript
        - Library
      - UserTS
```

`BackendLayouts`
: Hier werden alle definierten Backend-Layouts abgelegt, die der spätere Theme-Benutzer verwenden kann. Wie Sie diese erstellen und bereitstellen können, zeigen wir Ihnen in Kapitel 8.

`Elements`
: Hier werden alle definierten Inhaltselemente abgelegt. Die genaue Verzeichnisstruktur innerhalb dieses Verzeichnisses werden wir in Kapitel 10 behandeln.

`PageTS`
: Hier werden, wie soeben beschrieben, alle Seiten-TypoScript-Dateien abgelegt.

`TCA`
: Sofern es von Ihrem Theme benötigt wird, können Sie in diesem Verzeichnis spezielle PHP-Dateien anlegen, in denen das Table-Configuration-Array (kurz TCA) angepasst werden kann, um gegebenenfalls Formularfelder im Backend anzupassen.

`TypoScript`
: In diesem Verzeichnis werden, wie soeben beschrieben, die benötigten TypoScript-Dateien abgelegt.

`UserTS`
: Hier werden alle Benutzer-TypoScript-Dateien abgelegt, mit denen Sie die Einstellungen für Benutzer oder Benutzergruppen anpassen können. Diese Dateien sollten die Dateiendung `.userts` bekommen.

Erstellen Sie nun zunächst die folgenden Verzeichnisse und Dateien (siehe Beispiel 7-4) ohne Inhalt – in Kapitel 9 werden Sie dann erfahren, wie Sie diese mit Leben füllen.

Beispiel 7-4: Minimale Verzeichnisstruktur im Configuration-Verzeichnis

```
typo3conf
- ext
  - theme_bootstrap
    - Configuration
      - PageTS
        - Library
        - tsconfig.txt
      - TypoScript
        - Library
        - constants.txt
        - setup.txt
```

Das Verzeichnis Meta

Das *Meta*-Verzeichnis enthält, wie der Name bereits verrät, Metadaten zu Ihrem Theme, die später von der Erweiterung THEMES benötigt werden, um dem Benutzer alle notwendigen Informationen zum Theme bereitzustellen. Hierin muss eine *theme.yaml*-Datei enthalten sein, die die grundlegenden Metadaten in einer definierten Struktur enthält. Neben vielen optionalen Informationen sind es der Titel und die Beschreibung des Themes, die mindestens enthalten sein müssen.

Legen Sie nun das Verzeichnis *Meta* an und erstellen Sie darin eine Datei namens *theme.yaml*, die mindestens den folgenden Inhalt erhält (siehe Beispiel 7-5):

Beispiel 7-5: Pflichtangaben in der theme.yaml

```
# Name and description of the theme (required) ❶
title            : Minimalistic theme for reference purposes ❷
description      : | ❸
  This is the description of the theme. ❹
  With the starting pipe it can contain multiple lines.

  The newlines in this description will be preserved.
```

❶ Wenn eine Zeile mit einem # beginnt, wird sie als Kommentar ausgewertet.

❷ Einträge in einer YAML-Datei beginnen mit einem Bezeichner, gefolgt von einem Doppelpunkt. Hinter dem Doppelpunkt kommt dann der Wert des Eintrags. Hier haben wir einen Eintrag names `title`, der den Wert `Minimalistic theme for reference purposes` zugewiesen bekommen hat – also den Titel für unser Theme.

❸ Der Eintrag `description` beginnt genauso wie der Eintrag `title`, jedoch mit dem Unterschied, dass nach dem Doppelpunkt direkt eine sogenannte *Pipe* | folgt. Dies bedeutet in YAML, dass nun ein mehrzeiliger Wert kommt – die Beschreibung.

❹ Der Text dieses mehrzeiligen Werts muss jetzt mit zwei Leerzeichen eingerückt werden. Solange die Zeilen mit diesen zwei Leerzeichen eingerückt sind, ordnet YAML sie weiterhin dem mehrzeiligen Wert zu.

Als Nächstes sollten Sie sich als Theme-Autor in die `theme.yaml` eintragen (siehe Beispiel 7-6). Wenn Sie mit mehreren Autoren an einem Theme arbeiten – kein Problem, auch die Definition von mehreren Autoren ist möglich.

Beispiel 7-6: Definition von Autoren in der theme.yaml

```
# Name of the authors (optional, multiple possible)
authors: ❶
  - ❷
    name           : Thomas Deuling ❸
    email          : typo3@coding.ms
    company        : coding.ms
    companyLogo    :
    companyWebsite : www.coding.ms
  - ❹
    name           : Kay Strobach
    email          : typo3themes@kay-strobach.de
    company        : Kay Strobach
    companyLogo    :
    companyWebsite : www.kay-strobach.de
  -
    name           : Jo Hasenau
    email          : info@cybercraft.de
    company        : Cybercraft
    companyLogo    :
    companyWebsite : www.cybercraft.de
```

❶ Der Eintrag `authors` für die Angabe der Autoren wird genau so definiert wie der `title`-Eintrag ...

❷ ... nur mit dem Unterschied, dass eine Zeile mit zwei Leerzeichen und einem - (Bindestrich) folgt. Das bedeutet in YAML, dass es mehrere Werte zu diesem Eintrag geben kann – vergleichbar mit einem Array in Programmiersprachen.

❸ Die Definition der Werte von zum Beispiel Name oder E-Mail erfolgt wie gehabt und ist selbst erklärend.

❹ Wenn der Eintrag komplett ist, kann durch eine weitere Zeile mit zwei Leerzeichen und einem - (Bindestrich) ein neuer Eintrag gestartet werden.

Wichtig ist auch, dass Sie während der Theme-Entwicklung festhalten, welche Konstantengruppen Sie für die TypoScript-Konfiguration verwenden. Wie das im Detail funktioniert, werden Sie in Kapitel 9 kennenlernen – hier wollen wir nur kurz auf die Definition in der *theme.yaml* eingehen (siehe Beispiel 7-7).

Beispiel 7-7: Definition von Konstantengruppen in der theme.yaml

```
# Constants categories which are used for constants in this theme
constants: ❶
  availableCategories:  [theme,colors,page, ...] ❷
```

❶ Zuerst müssen Sie einen Eintrag `constants` anlegen, der einen Eintrag `availableCategories` enthält.

❷ Der Wert der `availableCategories` enthält nun in eckigen Klammern eine Auflistung der Konstantengruppen – diese sind einfach durch ein Komma getrennt. Welche Gruppen es hier gibt und welche Werte sie enthalten, werden Sie in Kapitel 9 kennenlernen.

Als Letztes wollen wir noch kurz zeigen, wie Sie definieren können, ob Ihr Theme eine bestimmte TYPO3-Erweiterung unterstützt. Das könnte zum Beispiel eine News-Erweiterung oder auch ein Log-in-Formular sein, für die Sie speziell angepasste Fluid-Templates oder sogar eine TypoScript-Konfiguration bereitstellen (siehe Beispiel 7-8). Wie Sie eine solche Erweiterung in Ihr Theme integrieren können, werden wir in Kapitel 16 näher betrachten.

Beispiel 7-8: Definition von unterstützten Erweiterungen in der theme.yaml

```
supportedExtensions: ❶
  news:
    version        : 1.0.0 ❷
    description    : News-Erweiterung, Templates auf Bootstrap-Basis ❸
  felogin: ❹
    version        : 6.2.0
    description    : Frontend-Userlogin, Template für Login-Formular
```

❶ Als Erstes wird ein Eintrag namens `supportedExtensions` erstellt, der beliebig viele Untereinträge erhalten darf. Der erste Untereintrag ist hier die `news`-Erweiterung.

❷ Zu jedem Eintrag sollte nun die Version vermerkt werden, für die Sie die Integration vorgenommen haben.

❸ Des Weiteren sollte auch eine Beschreibung angegeben werden, zum Beispiel darüber, in welcher Art die Integration vorgenommen wurde oder auch welche Templates integriert wurden.

❹ Ein Eintrag für eine weitere Erweiterung wie hier `felogin` würde wieder nach dem gleichen Schema erfolgen.

Aber auch zusätzliche Informationen wie zum Beispiel Lizenzen, Schlüsselwörter oder ein Link zu einer Demoversion können hier angegeben werden.

Eine komplette Auflistung der möglichen enthaltenen Informationen können Sie unter *typo3-themes.org/meta-theme-yaml/* nachlesen.

Das *Meta*-Verzeichnis enthält ein weiteres Unterverzeichnis namens *Screenshots*, in dem neben einer *screenshot.png*-Datei (die möglichst vorhanden sein sollte und als Vorschaubild für den Theme-Selektor verwendet wird) beliebig viele weitere Vorschaubilder des Themes abgelegt werden können. Die Definition der weiteren Vorschaubilder erfolgt ebenfalls über die Datei *theme.yaml*, sodass Sie zu jedem Bild auch einen kurzen Text angeben können (siehe Beispiel 7-9).

Beispiel 7-9: Definition von Vorschaubildern in der theme.yaml

```
# Screenshots for the preview gallery (optional, multiple possible)
screenshots: ❶
  -
    file           : screenshot-01.png ❷
    caption        : Startseite mit Slider ❸
  -
    file           : screenshot-02.png
    caption        : Kontakt-Seite
```

❶ Im Eintrag `screenshots` können wieder beliebig viele Untereinträge erstellt werden.

❷ Im Eintrag `file` muss der Dateiname inklusive Dateierweiterung angegeben werden.

❸ Im Eintrag `caption` können Sie nun eine kurze Beschreibung zum Bild hinterlegen.

Wenn Sie bereits ein grafisches Layout Ihres Themes vorliegen haben, erstellen Sie nun davon einige Vorschaubilder und legen diese wie beschrieben im *Meta/Screenshots*-Verzeichnis ab.

Abschließend sollte Ihr *Meta*-Verzeichnis wie folgt aussehen (siehe Beispiel 7-10):

Beispiel 7-10: Meta-Verzeichnis

```
typo3conf
- ext
  - theme_bootstrap
    - Meta
```

```
    - Screenshots
      - screenshot.png
      - screenshot-01.png
      - screenshot-02.png
    - theme.yaml
```

Das Verzeichnis Resources

Bei den Ressourcen eines Themes, genauso wie bei TYPO3-Erweiterungen oder Site-Packages für TYPO3 Neos, unterscheiden wir zwischen *Private Resources* und *Public Resources*. Somit müssen Sie unter dem Verzeichnis *Resources* ein Verzeichnis namens *Private* und eines namens *Public* erstellen.

Private Resources
: Unterhalb des *Private*-Verzeichnisses liegen alle Dateien, auf die nicht direkt zugegriffen werden darf – die also privat sind. Dies können Fluid-Templates, Sprachdateien oder auch CSS-Basisdateien für die Erzeugung von dynamischem CSS sein.

Public Resources
: Unterhalb des *Public*-Verzeichnisses liegen alle Dateien, auf die der Browser direkt zugreifen darf. Dies sind beispielsweise Bilder, JavaScript- oder CSS-Dateien.

Damit wirklich niemand von außen auf die Dateien im *Private*-Verzeichnis zugreifen kann, erstellen Sie nun eine Textdatei mit dem Namen *.htaccess* in diesem Verzeichnis. Diese Datei wird von Ihrem Webserver ausgewertet und muss die folgende Zeile enthalten (siehe Beispiel 7-11), damit ein Zugriff von außen nicht möglich ist.

Beispiel 7-11: Zugriffsschutz mit einer .htaccess-Datei

```
deny from all
```

Werfen Sie als Nächstes einen Blick auf die Verzeichnisse, die in den *Private Resources* vorhanden sein können, und erstellen Sie diese Verzeichnisse ebenfalls in Ihrem Theme.

Beispiel 7-12: Verzeichnisstruktur der privaten Ressourcen

```
typo3conf
- ext
  - theme_bootstrap
    - Resources
      - Private
        - Dyncss
          - Library
        - Extensions
        - Fonts
        - Layouts
        - Language
        - Partials
        - Templates
        - .htaccess
```

`Dyncss`
: Hier werden alle DynCSS-Dateien abgelegt. Dies sind, je nachdem, welchen DynCSS-Adapter man verwendet, LESS-, SASS- oder beispielsweise Turbine-Dateien. Auch hier sollte wieder ein *Library*-Verzeichnis erstellt werden, in dem die Stile in Dateien nach Elementen oder Bereichen gekapselt abgelegt werden sollten. Mehr dazu aber in Kapitel 12.

`Extensions`
: Wenn das Theme weitere Erweiterungen (sprich Extensions) wie zum Beispiel die News-Erweiterung unterstützt, werden die zugehörigen *privaten* Dateien in diesem Verzeichnis abgelegt. In welcher Struktur das geschehen sollte, wird in Kapitel 16 behandelt.

`Fonts`
: Hier werden alle Fonts abgelegt, die nur serverseitig, beispielsweise für die Erstellung von Grafiken mit dem GIFBUILDER oder für die PDF-Generierung, verwendet werden.

`Layouts`
: Hier legen Sie alle Fluid-Layouts ab.

`Language`
: Legen Sie hier alle Sprachdateien ab, die die sprachenabhängigen Labels des Themes enthalten.

`Partials`
: Dies ist die Ablage für alle Fluid-Partials.

`Templates`
: Legen Sie hier alle Fluid-Templates ab.

Natürlich können an dieser Stelle je nach Umfang des Themes auch weitere Verzeichnisse existieren. Wenn zum Beispiel Bilder für die serverseitige Verarbeitung benötigt werden, könnte es auch ein `Images`-Verzeichnis geben.

Im Verzeichnis *Public Resources* können die folgenden Unterverzeichnisse vorhanden sein (siehe Beispiel 7-13). Erstellen Sie auch diese Verzeichnisse nach Bedarf in Ihrem Theme.

Beispiel 7-13: Verzeichnisstruktur der öffentlichen Ressourcen

```
typo3conf
- ext
  - theme_bootstrap
    - Resources
      - Public
        - Contrib
        - Extensions
        - Fonts
        - Icons
        - Images
        - JavaScript
        - Stylesheets
```

`Contrib`
: Hier können Third-Party-Bibliotheken abgelegt werden. Wenn Sie beispielsweise Bootstrap nicht über ein CDN einbinden wollen, sondern es direkt mit Ihrem Theme mitliefern möchten, erstellen Sie hier ein entsprechendes Verzeichnis und legen den Quellcode von Bootstrap darin ab.

`Extensions`
: Hier legen Sie, sofern Sie weitere Erweiterungen verwenden, die *public*-Dateien dieser Erweiterungen ab. Darauf werden wir ebenfalls weiter unten in Kapitel 16 näher eingehen.

`Fonts`
: Legen Sie hier die Font-Dateien ab, die im Theme verwendet werden.

`Icons`
: Alle Icons des Themes werden hier abgelegt.

`Images`
: Dies ist die Ablage für alle Bilder des Themes. Um eine bessere Übersicht zu behalten, können hier auch weitere Unterverzeichnisse erstellt werden.

`JavaScript`
: Hier legen Sie alle JavaScript-Dateien ab, die speziell für dieses Theme entwickelt wurden. Fremde Skripte sollten besser unter dem Verzeichnis *Contrib* abgelegt werden.

`Stylesheets`
: Alle statischen CSS-Dateien werden hier abgelegt.

Theme-Konfigurationsdateien

Da Sie Ihr Theme als eigenständige Erweiterung bereitstellen wollen, benötigen Sie dazu eine entsprechende Erweiterungskonfiguration, damit das Theme später vom Erweiterungsmanager und auch von der THEMES-Erweiterung gelesen werden kann. Was Sie für eine solche Konfiguration konkret benötigen, erläutern wir Ihnen in diesem Abschnitt.

Die Konfiguration des Erweiterungsmanagers wird laut Konvention in einer Datei *ext_emconf.php* im Wurzelverzeichnis der Theme-Erweiterung platziert. Diese PHP-Datei enthält ein definiertes PHP-Array, in dem alle wichtigen Informationen und Eigenschaften der Theme-Erweiterung definiert sind. Eine solche Datei könnte nun wie folgt aussehen (siehe Beispiel 7-14) – erstellen Sie eine eigene *ext_emconf.php* im Wurzelverzeichnis Ihres Themes und setzen Sie diesen PHP-Quellcode darin ein.

Beispiel 7-14: Beispiel für eine Erweiterungsmananger-Konfiguration

```
<?php
$EM_CONF[$_EXTKEY] = array(
  'title' => 'Bootstrap Theme',
  'description' => 'Theme to use Twitter Bootstrap',
  'category' => 'templates',
  'shy' => 0,
```

```
  'version' => '1.0.0',
  'dependencies' => '',
  'conflicts' => '',
  'priority' => '',
  'loadOrder' => '',
  'module' => '',
  'state' => 'beta',
  'uploadfolder' => 0,
  'createDirs' => '',
  'modify_tables' => '',
  'clearcacheonload' => 0,
  'lockType' => '',
  'author' => 'Themes-Team',
  'author_email' => 'team@typo3-themes.org',
  'author_company' => '',
  'CGLcompliance' => '',
  'CGLcompliance_note' => '',
  'constraints' => array(
    'depends' => array(
      'typo3'  => '6.2.0-6.2.99',
    ),
    'conflicts' => array(
    ),
    'suggests' => array(
    ),
  ),
  '_md5_values_when_last_written' => 'a:0:{}',
);
?>
```

Im Folgenden wollen wir nun kurz auf die wichtigsten Eigenschaften eingehen, die Sie dann umgehend in Ihrer Kopie dieser Datei anpassen sollten.

`title`
: Hier sollte der Titel bzw. Name des Themes eingetragen werden. Dieser wird im Erweiterungsmanager und für die Selektion in der THEMES-Erweiterung als Default-Titel verwendet. Er kann jedoch in der Datei *Meta/theme.yaml* für den Themes-Selektor überschrieben werden.

`description`
: Geben Sie hier eine kurze Beschreibung des Themes an.

`category`
: Hier tragen Sie die Kategorie der Erweiterung ein. Da Sie mit einem Theme im Prinzip eine Frontend-Vorlage entwickeln, ist das in diesem Fall `templates`.

`version`
: Wenn ein Theme stetig weiterentwickelt wird, sollten seine Entwicklungsschritte mit entsprechenden Versionsnummern in diesem Feld vermerkt werden.

`author`
: Hier tragen Sie den Namen des bzw. der Theme-Entwickler ein.

`author_email`
: Hier sollte eine E-Mail-Adresse angegeben werden, sodass eine Kontaktaufnahme mit den Entwicklern möglich ist.

`constraints`
: Das Feld `constraints` zeigt Abhängigkeiten bzw. Konflikte zu TYPO3 oder darin installierten Erweiterungen an.

Nachdem Sie nun wissen, welche grundlegende Eigenschaften Sie in der *ext_emconf.php*-Datei benötigen, werden wir noch einmal kurz auf die Theme-Erweiterung sowie Abhängigkeiten bzw. mögliche Konflikte zu anderen Erweiterungen eingehen. Dafür wird das Feld `constraints` in die folgenden drei Bereiche aufgeteilt:

`depends`
: `depends` listet Erweiterungen auf, die für eine fehlerfreie Ausführung dieser Theme-Erweiterung benötigt werden.

`conflicts`
: `conflicts` listet Erweiterungen auf, die mit dieser Theme-Erweiterung nicht verwendet werden sollten, da es sonst zu Fehlern kommen kann.

`suggests`
: `suggests` listet Erweiterungen auf, die mit dieser Theme-Erweiterung funktionieren und deren Installation empfohlen wird.

Diese `constraints` können Sie ganz einfach definieren, indem Sie als Array-Key den jeweiligen Erweiterungsschlüssel verwenden und als Array-Value für diesen Key den zulässigen Versionsbereich angeben. Eine Definition für die Abhängigkeit von TYPO3 (mindestens Version 6.2, maximal 6.2.99) würde wie folgt aussehen.

Beispiel 7-15: Abhängigkeiten der Theme-Erweiterung

```
<?php
$EM_CONF[$_EXTKEY] = array(
  ...
  'constraints' => array( ❶
    'depends' => array( ❷
      'typo3'  => '6.2.0-6.2.99', ❸
      'themes' => '1.0.1-1.0.1', ❹
      'themes_gridelements' => '0.1.0-1.0.0', ❺
    ),
    'conflicts' => array(
    ),
    'suggests' => array(
    ),
  ),
  ...
);
```

❶ Zur Angabe der Abhängigkeiten des Themes suchen Sie den `constraints`-Knoten.

❷ Hier wird nun die Abhängigkeit zu TYPO3, der THEMES-Erweiterung und zu den Basisbibliotheken von *Gridelements* definiert.

❸ Als Erstes geben Sie an, dass Sie TYPO3 zwischen Version 6.2.0 und 6.2.99 erwarten.

❹ Da das Theme nur zusammen mit der THEMES-Erweiterung funktionieren wird, geben Sie auch diese hier entsprechend an.

❺ Abschließend geben Sie noch die `themes_gridelements`-Erweiterung an, da wir auf deren Bibliotheken aufsetzen.

Für weitere Informationen zur Erweiterungsmanager-Konfiguration schauen Sie einfach in die EM-Conf-Dokumentation.

Die komplette Dokumentation zur Erweiterungsmanager-Konfiguraion (EM-Conf) ist unter dem folgenden Link zu finden: *http://docs.typo3.org/typo3cms/CoreApiReference/ExtensionArchitecture/DeclarationFile/Index.html*

Als Letztes benötigen Sie noch ein Icon für Ihre Themes-Erweiterung, das dann in der Datei *ext_icon.gif* abgelegt wird. Dieses Icon wird im Erweiterungsmanager als Symbol verwendet und sollte 16 Pixel breit und 16 Pixel hoch sein.

Abschließend sollte gesagt werden, dass Ihre Themes-Erweiterung im Backend installiert werden muss, damit Sie vollen Zugriff auf die Themes-Dateien erhalten – anderfalls wird Ihr Theme nicht fehlerfrei funktionieren. Dies kann wie gewohnt über den Erweiterungsmanager geschehen.