Index.php - Dokumentation (Joomla Template)

Dieses Dokument beschreibt Aufbau, Zweck und empfohlene Struktur der Template-Datei *index.php* des Joomla-Templates **joomla-tpl-prosoft** und dient als technische Referenz für den Seitenaufbau, die dynamische CSS-Generierung sowie die Integration von Template-Funktionen und Modulpositionen.

1. Dateien-Überblick

index.php

2. Initialisierung, Kontext und Assets

In diesem Abschnitt wird beschrieben, wie das Template im Frontend initialisiert wird und welche grundlegenden Informationen und Ressourcen dafür benötigt werden.

2.1 Initialisierung und Kontext

Zu Beginn wird das Joomla-Application-Object sowie Input und Asset-Manager initialisiert:

- $app = Factory::getApplication(); → Zugriff auf Joomla-Kontext
- $input = $app->getInput(); → Zugriff auf Request-Parameter
- $webAssetManager = $this->getWebAssetManager(); → Joomla 5 WebAssetManager für CSS und JavaScript

Zusätzlich werden Template-relevante Seitenparameter ermittelt:

- option, view, layout, task, itemid (aus dem Request)
- pageclass_sfx wird über das aktive Menü-Item geladen (Workaround, da getMenu() nur im Frontend existiert)

Diese Werte werden später für dynamische Body-Classes verwendet (z.B. com_content view-article …), was gezieltes Styling pro Ansicht ermöglicht.

2.2 Website Icon

Die *index.php* unterstützt ein konfigurierbares Website-Icon über den Template-Parameter *websiteIcon*:

- wenn *websiteIcon* gesetzt ist, wird daraus eine absolute URL gebaut (Uri:root(true) + rel. Pfad)
- der Dateityp wird über die Dateiendung ermittelt und als type-Attribut im <link rel="icon"> gesetzt (z.B. image/png, image/svg+xml, usw.)
- zusätzlich wird ein Versionsparameter !v= angehängt:
    - wenn die Datei lokal existiert: filemtime() (Änderungszeitpunkt)
    - sonst Fallback: time()

Ergebnis:

Änderungen am Icon innerhalb von Joomla werden zuverlässig vom Browser neu geladen.

Fallback:

Wenn kein gültiges Icon vorliegt, werden Joomla-Standard-Favicons eingebunden:

- favicon.ico (alternatives Icon)
- joomla-favicon-pinned.svg (mask-icon, mit Farbe)

2.2 Meta-Setup

Die Datei setzt ein viewport-Meta-Tag für responsive Darstellung:

- viewport: width=device-width, initial-scale=1

2.3 Asset-Einbindung über Joomla WebAssetManager

Das Template nutzt das Joomla-Asset-System (Joomla-5-Standard):

2.3.1 Core-Asset

- $webAssetManager->useStyle(‚bootstrap.css'); → bootstrap.css wird aus dem Joomla-Core geladen

2.3.2 Template-Assets (aus joomla.asset.json)

Es werden mehrere registrierte Assets eingebunden (Scripts & Styles). Wichtig: die Strings wie *joomla-tpl-prosoft-template.script* sind **Handles**, die in *joomla.asset.json* definiert sind.

JavaScript:

- joomla-tpl-prosoft-template.script

- joomla-tpl-prosoft-font.script
- joomla-tpl-prosoft-darkmode.script
- joomla-tpl-prosoft-mobile.script

CSS:

- joomla-tpl-prosoft.styles
- joomla-tpl-prosoft.hero
- joomla-tpl-prosoft.popup
- joomla-tpl-prosoft.custom

2.4 Sticky Header

Über den Template-Parameter stickyHeader wird optional eine Sticky-Klasse gesetzt, welche bei aktiv *position-sticky sticky-top* heißt.

2.5 Custom CSS

Der Parameter *customCSS* wird aus den Template-Optionen gelesen und später im<head> als Inline-<style> ausgegeben.

3. Dynamische CSS-Generierung

Die Datei *index.php* ist für die **Einbindung und Steuerung** der dynamischen CSS-Generierung verantwortlich, erzeugt das CSS jedoch **nicht selbst**. Stattdessen greift sie auf externe Hilfsdateien zurück, die gemeinsam die parametergesteuerte Gestaltung des Templates ermöglichen.

3.1 Beteiligte Dateien

- index.php → Orchestrierung: Aufruf des Generators, Schreiben der Dateien, Einbindung ins Dokument
- tpl_css_generator.php → erzeugt CSS und Hash auf Basis der Template-Parameter
- variables_skip.php → definiert Parameter, die nicht automatisch als CSS-Variablen ausgegeben werden
- templateDetails.xml → Quelle der Template-Parameter (Backend-Konfiguration)

3.2 Rolle der index.php im Detail

Die *index.php* übernimmt innerhalb der dynamischen CSS-Generierung eine **koordinierende Rolle**. Sie ruft den externen CSS-Generator auf, verwaltet die Ausgabe der generierten Dateien und bindet das resultierende Stylesheet in das HTML-Dokument ein.

3.2.1 Aufruf des Generators

Die *index.php* lädt den CSS-Generator und erhält ein Ergebnis-Array:

- css → vollständiger CSS-Code
- hash → eindeutiger Hash des CSS-Zustands

Damit bleibt die eigentliche CSS-Logik ausgelagert, während die *index.php* nur die Integration übernimmt und performant bleibt.

3.2.2 Schreiben der generierten Daten

Die *index.php*:

- legt den Zielordner im Joomla-Media-Verzeichnis an
- schreibt *generated.css* und *generated.hash*
- vermeidet unnötige Schreibzugriffe durch Hash-Vergleich

Dies erhöht die Performance immens und reduziert Dateizugriffe.

3.2.3 Cache-Handling

Nach einer Änderung:

- wird das CSS mit einer Hash-basierten Versionsnummer eingebunden
- bei Administration zusätzlich der Joomla-System- und Page-Class geleert

Änderungen an Template-Parametern werden dadurch sofort sichtbar.

3.2.4 Einbettung ins HTML-Dokument

Das generierte Stylesheet wird wie eine normale CSS-Datei eingebunden und ergänzt die statischen Template-Styles. Optionales benutzerdefiniertes CSS aus den Template-Optionen wird zusätzlich inline ausgegeben.

3.3 Dokumentationshinweis

Dieser Abschnitt beschreibt bewusst **nur die Rolle der** *index.php* innerhalb der CSS-Generierung. Die interne Funktionsweise des CSS-Generators selbst ist kein Bestandteil dieser Datei und wird daher nicht im Detail behandelt.

4. Body-Struktur und Feature-Blöcke

Dieser Abschnitt beschreibt den strukturellen Aufbau des HTML-Bodys innerhalb der *index.php* sowie die bedingte Ausgabe der einzelnen Layout- und Feature-Blöcke. Der Fokus liegt auf der modularen Darstellung von Header, Navigation, Hero, Content, Sidebars, Footer und optionalen Template-Funktionen.

4.1 Body-Class-System

Die body-Klasse wird dynamisch aus Request-Infos zusammengesetzt:

- option
- view-*
- Optional layout-*
- Optional task-*
- Optional itemid-*
- Optional pageclass_sfx aus dem aktiven Menüpunkt

4.2 Popup (optional)

Der Popup-Block wird nur gerendert, wenn der Template-Parameter aktiv ist:

- Bedingung: $this->params->get(‚usePopup‘ === ‚system‘
- Ausgabe als Overlay (popup-overlay) + Dialog-Container (role="dialog", aria-live="polite")

Inhalte:

- Optionales Bild (wenn gesetzt)
- Heading, Text
- Button (z.B. „Fortfahren")

4.3 Social Bar (optional)

Die Social-Bar wird nur gerendert, wenn der Template-Parameter aktiv und Daten vorhanden sind:

- Bedingung: !empty($useSocialBar) && !empty($socialItems)

Funktionsweise:

- Erlaubt mehrere Social-Links (Instagram, Facebook, LinkedIn, YouTube, X, Mail)
- Icon-Quelle ist ein lokaler SVG-Pfad im Media-Ordner
- SVG wird inline geladen (per Helper-Funktion $inlineSvg())

Sicherheit & UX:

- Externe Links: target="_blank" rel="noopener"
- aria-label je Link

4.4 Seiten-Wrapper

```
<div class="site-container">
        <div class="site-container-shadow">
                <header> … </header>
                (Mobile Menu) <?php if( … ) ?> … <?php endif; ?>
                (Hero) <?php if( … ) ?> … <?php endif; ?>
                (Scroll-to-top-Button) <?php if( … ) … <?php endif; ?>
                <div class ="content-wrap" <?php echo $layoutClass;?>"> … </div>
                <footer> …  </footer>
        </div>
</div>
```

4.5 Header-Struktur

Der Header besteht aus header-left + header-right und ist optional sticky.

Branding:

- Bedingung: $this->params->get(‚brand', 1)
- Links: menu-top (optional) + logo-left + menu-bottom (optional)
- Rechts: logo-right (optional)

Header-Module:

- Contact (oben)
- Menu, menu-top, menu-bottom (Navigation)
- Search
- Optionen: Font-Size-Buttons + Dark-Mode-Button (je nach Parameter)

4.6 Mobile Navigation

Wenn relevante Module belegt sind, wird Mobile-Navigation gerendert:

- Burger-Button
- <nav class="mobile-nav" …>
- Close-Button
- Optional search / menu / menu-top / menu-bottom
- Unten optional: Font-Size-Button / Dark-Mode-Button / contact
- Overlay für UX

4.7 Hero-Bereich

Es gibt 3 Fälle:

1. Hero-Datei + Hero-Modul → beide werden gerendert (hero-both)
2. Hero-Datei → Datei wird gerendert (hero-file)
3. Hero-Modul → Modul wird gerendert (hero-module)

Zusätzlich kann das welcome-Modul in allen Fällen darübergelegt werden.

4.8 Scroll-to-top-Button (optional)

- Bedingung: $this->params->get(‚useScrollToTopButton') === ‚system'
- Button ruft JavaScript-Funktion backToTop() auf
- Icon wird dynamisch ausgegeben

4.9 Content und Sidebars

Layout wird über $layoutClass gesteuert:

- Sidebar links: sidebar-left (optional)
- Main Content: message + component
- Zusatzmodule: component-top und component-bottom
- Sidebar rechts: sidebar-right (optional)

4.10 Footer und Breadcrumb

Footer zeigt bis zu vier Spalten: footer-a, footer-b, footer-c, footer-d (jeweils optional)

Darunter: breadcrumb (optional)