TemplateDetails.xml - Dokumentation (Joomla Template)

Dieses Dokument beschreibt Aufbau, Zweck und empfohlene Struktur der Manifest-Datei *templateDetails.xml* des Joomla-Templates **joomla-tpl-prosoft** und dient als technische Referenz für die Konfiguration und Architektur des Templates.

1. Dateien-Überblick

templateDetails.xml

2. Grundlegender Aufbau

<?xml?>-Element:

- Definition der XML-Version und Encoding (i.d.R. UTF-8)
- Typische-Attribute:
  - version = „1.0"
  - encoding = „UTF-8"

<extension>-Element:

- Root-Element beschreibt das Template-Manifest
- Typische Attribute:
  - type = „template" → Joomla Template-Erweiterung
  - client = „site" → Frontend-Template
  - method = „upgrade" → ermöglicht Upgrade-Installationen, ohne vorherige Deinstallation
- Child-Attribute:
  - <name>
  - <version>
  - <creationDate>
  - <author>
  - <authorEmail>
  - <description>
  - <inheritable>
  - <files> → Template-Dateien / Ordner, die installiert werden
  - <media> → Assets (CSS / JS / Images) im Media-Ordner
  - <positions> → definierte Modulpositionen
  - <config> → Template-Optionen (Parameter) im Backend

Minimales Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<extension type="template" client="site" method="upgrade">
    <name>joomla-tpl-prosoft</name>
    <version>1.0</version>
    <creationDate>04.11.2025</creationDate>
    <author>Franz Hielscher</author>
    <authorEmail>franz.hielscher@prosoft-krippner.com</authorEmail>
    <copyright>(C) 2026 ProSoft Krippner GmbH</copyright>
    <description>Joomla Template - ProSoft Krippner GmbH</description>
    <inheritable>1</inheritable>
    <files> … </files>
    <media destination="templates/site/joomla-tpl-prosoft" folder="media"> … </media>
    <positions> … </positions>
    <config> … </config>
</extension>
```

3. Metadaten-Elemente

Die Metadaten beschreiben Identität, Herkunft und Zweck des Templates. Sie werden von Joomla u.a. im Backend, beim Installieren und bei Updates verwendet.

3.1 Pflicht-Metadaten

- <name> → Technischer Name des Templates (Ordnername)
- <version> → Aktuelle Template-Version
- <creationDate> → Erstellungsdatum (TT.MM.JJJJ)
- <author> → Name des Entwicklers / der verantwortlichen Person
- <description> → Kurzbeschreibung des Templates im Joomla-Backend

3.1.1 Hinweis zur Versionierung

Die <version> sollte bei **jeder funktionalen Änderung** erhöht werden. In Verbindung mit method = „upgrade" ermöglicht dies saubere Template-Updates.

3.1.2 Beispiel

```
<name>joomla-tpl-prosoft</name>
<version>1.0</version>
<creationDate>04.11.2025</creationDate>
<author>Franz Hielscher</author>
<description>Joomla Template - ProSoft Krippner GmbH</description>
```

3.2 Empfohlene Metadaten

- <authorEmail> → Kontaktadresse für Wartung und Support
- <inheritable> → Erlaubt Child-Templates (1 = aktiv)

3.2.1 Beispiel

```
<authorEmail>franz.hielscher@prosoft-krippner.com</authorEmail>
<inheritable>1</inheritable>
```

4. Dateien & Media-Strukturen

Das <files>-Element definiert alle Dateien, die im Template-Root-Verzeichnis installiert werden. Im ProSoft-Template sind dies neben der eigentlichen Template-Datei auch Hilfs- und Konfigurationsdateien.

4.1 <files> - Template-Dateien

Das <files>-Element definiert alle Dateien und Ordner, die im Template-Verzeichnis installiert werden.

- index.php → Einstiegspunkt des Templates, steuert das Frontend-Rendering
- joomla.asset.json → Definition von CSS- und JavaScript-Assets nach Joomla-Standard
- templateDetails.xml → Manifest- und Konfigurationsdatei
- tpl_css_generator → Dynamische Generierung von CSS basierend auf Template-Parametern
- variables_skip.php → Definition von Variablen, die von der CSS-Generierung ausgeschlossen werden
- html/ → Template-Overrides für Joomla-Komponenten und Module

4.1.1 Beispiel

```
<files>
  <filename>index.php</filename>
  <filename>joomla.asset.json</filename>
  <filename>templateDetails.xml</filename>
  <filename>tpl_css_generator.php</filename>
  <filename>variables_skip.php</filename>
  <folder>html</folder>
</files>
```

4.1.2 Besonderheit: Dynamische CSS-Generierung

Das ProSoft-Template nutzt eine **serverseitige CSS-Generierung**, um:

- Template-Parameter in CSS-Variablen zu überführen
- Redundante CSS-Dateien zu vermeiden
- Design-Änderungen ohne manuelle CSS-Anpassung zu ermöglichen

Die Steuerung erfolgt über:

- templateDetails.xml (Parameter)
- tpl_css_generator.php
- variables_skip.php

Folge: Enge Kopplung zwischen Template-Optionen und Frontend-Design.

4.2 <media> - Asset-Verzeichnis

Das <media>-Element definiert den Zielpfad für statische Assets im Joomla-Media-Ordner. Die dort abgelegten Dateien werden **nicht automatisch eingebunden**, sondern über das Joomla-Asset-Management geladen.

4.2.1 Beispiel

```
<media destination="templates/site/joomla-tpl-prosoft" folder="media">
  <folder>css</folder>
  <folder>images</folder>
  <folder>js</folder>
</media>
```

4.3 Trennung von Template-Logik und Assets (Best Practice)

Vorteile dieser Struktur:

- saubere Trennung von PHP und Assets
- bessere Updatefähigkeit
- klare Wartbarkeit
- Joomla-5-konform
- Kompatibel mit Caching und Asset-Management

5. Definition der Modulpositionen

Modulpositionen definieren die Bereiche des Templates, in denen Joomla-Module ausgegeben werden können. Sie werden im Backend zur Modulzuweisung verwendet und im Template (index.php) gerendert.

5.1 Beschreibung der Modulpositionen

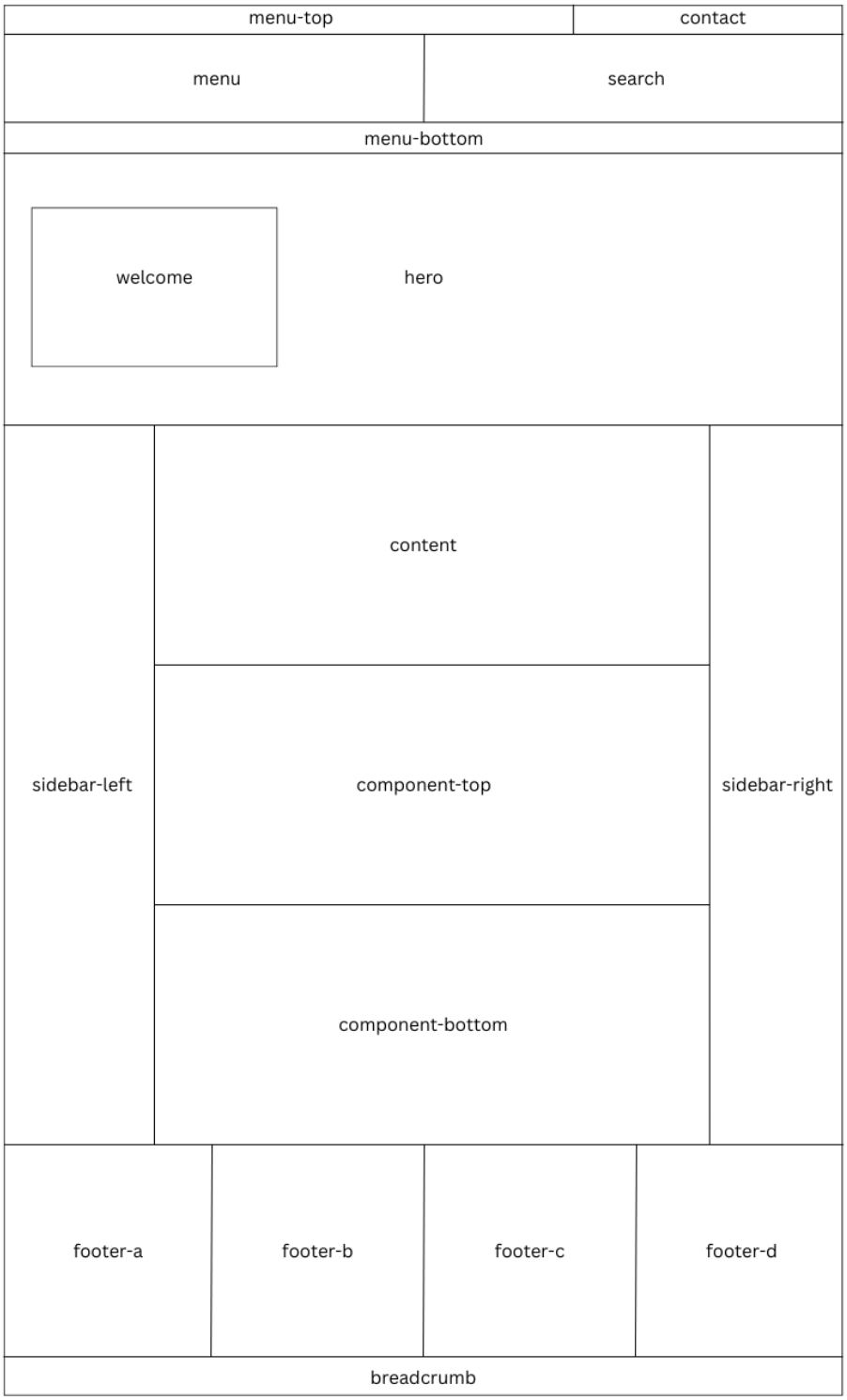
- menu-top → Module links oberhalb der Hauptnavigation
- menu → Hauptnavigation
- menu-bottom → Module links unterhalb der Hauptnavigation
- contact → Kontaktinformationen, CTA-Module
- search → Suchmodule
- hero → Hero-Bereich, Slider, Bilder, großer Teaser
- welcome → Begrüßung oder Einleitungsbereich
- component-top → Module unterhalb der Hauptkomponente
- component-bottom → Module unterhalb der component-top und der Hauptkomponente
- sidebar-left → Linke Seitenliste
- sidebar-right → Rechte Seitenliste
- footer-a → Footer-Spalte A
- footer-b → Footer-Spalte B
- footer-c → Footer-Spalte C
- footer-d → Footer-Spalte D
- breadcrumb → Breadcrumbs, Copyright, Navigation

5.1.1 Beispiel

```
<positions>
  <position>menu-top</position>
  <position>menu</position>
  <position>menu-bottom</position>
  <position>contact</position>
  <position>search</position>
  <position>hero</position>
  <position>welcome</position>
  <position>sidebar-left</position>
  <position>sidebar-right</position>
  <position>component-top</position>
  <position>component-bottom</position>
```

```
<position>footer-a</position>
  <position>footer-b</position>
  <position>footer-c</position>
  <position>footer-d</position>
  <position>breadcrumb</position>
</positions>
```

5.2 Template-Wireframe



5.3 Namenskonventionen

- durchgängige **Kleinbuchstaben**
- **kebab-case**
- semantisch und layoutbezogen

5.4 Hinweise zur Nutzung

- nicht jede Position muss auf jeder Seite belegt sein
- leere Positionen werden im Template nicht gerendert
- die Reihenfolge in der XML dient nur der Übersicht, nicht dem Layout

6. Template-Konfiguration

Das <config>-Element definiert alle Template-Parameter, die im Joomla-Backend konfiguriert werden können. Diese Parameter beeinflussen Layout, Design und Verhalten des Templates. Die Konfiguration ist in logisch getrennte Fieldsets unterteilt. Jedes Fieldset bündelt thematisch zusammengehörige Optionen und steuert einen bestimmten Bereich des Templates.

6.1 Beschreibung der Fieldsets

- General → Grundlayout, Branding (Logo, Titel), Abstände, Website-Breite, Header-Verhalten
- Menu → Navigation, Menüfarben, Dropdown-Design, Header-Hintergrund
- Hero → Hero Bild als Fallback, Welcome-Modul-Design, Einstiegsgestaltung
- Content → Breiten, Box-Shadows, Content-Farben, Buttons
- Social Bar → Aktivierung und Gestaltung der Social-Media-Leiste
- Sidebar → Breite, Farben, Sticky-Verhalten der Sidebars
- Scrollbar → Individuelle Gestaltung der Browser-Scrollbar
- Footer → Footer-Layout, Spaltengewichtung, Farben
- Font Scheme → Auswahl von System- oder hochgeladenen Schriftarten
- Font Weight → Steuerung der Schriftstärken einzelner Bereiche
- Popup → Konfigurierbares Overlay-Popup
- Scroll To Top Button → Button zum Zurückscrollen nach oben
- Font Size Button → Button zur Schriftgrößenanpassung
- Dark Mode → Umschaltbarer Dark Mode inklusive Farbdefinitionen
- Mobile → Mobile Navigation, Abstände, Farben

6.1.1 Beispiel

```
<fieldset name = "general" label = "General"> … </fieldset>
<fieldset name = "menu" label = "Menu"> … </fieldset>
<fieldset name = "hero" label = "Hero"> … </fieldset>
<fieldset name = "content" label = "Content"> … </fieldset>
<fieldset name = "socialBar" label = "Social Bar"> … </fieldset>
<fieldset name = "sidebar" label = "Sidebar"> … </fieldset>
<fieldset name = "scrollbar" label = "Scrollbar"> … </fieldset>
<fieldset name = "footer" label = "Footer"> … </fieldset>
<fieldset name = "fontScheme" label = "Font Scheme"> … </fieldset>
<fieldset name = "fontWeight" label = "Font Weight"> … </fieldset>
<fieldset name = "popup" label = "Popup"> … </fieldset>
<fieldset name = "scrollToTopButton" label = "Scroll To Top Button“> … </fieldset>
<fieldset name = "fontSizeButton" label = "Font Size Button"> … </fieldset>
<fieldset name = "darkMode" label = "Dark Mode"> … </fieldset>
<fieldset label="Mobile" name="mobile"> … </fieldset>
```

6.2 Funktionsprinzip

- Jedes Fieldset wird im Joomla-Backend als eigener Abschnitt dargestellt
- Viele Optionen werden kontextabhängig eingeblendet
- Design-relevante Parameter fließen in die dynamische CSS-Generierung
- Funktionale Schalter werden direkt im Template ausgewertet

6.3 Dokumentationsstrategie

Diese Dokumentation beschreibt **bewusst nur die Fieldsets**, nicht jedes einzelne Feld.

Gründe:

- Bessere Lesbarkeit
- Geringerer Wartungsaufwand
- Fokus auf Architektur statt Implementierungsdetails