Content-Block-Builder

Einleitung:

Der Content-Block-Builder – kurz „cb“ – ist inspiriert durch die bereits existierende Extension „Content Blocks“.

Während Grundkonzept und Resultat ähnlich bis gleich sind, gibt es dennoch einige Unterschiede im Detail.

Beide Extensions benutzen eine Yaml-Datei, mit dessen Hilfe man einen Content-Block erstellen kann.

Dabei legt cb den Fokus auf drei Aspekte:

1. Cb soll sowohl von Entwicklern, Integratoren als auch Redakteuren verwendet werden können.
2. Cb erstellt alle nötigen Dateien automatisch anhand der Einträge in der Yaml-Datei. Dadurch lassen sich auch unabhängig von der Extension Änderungen vornehmen. Einstellungen lassen sich auch zwischen der Yaml- und den PHP-Dateien mischen. Mehr dazu später.
3. Cb bietet ein komplexes Error-Handling um saubereren Code zu schreiben, Tippfehler zu vermeiden und eventuell eine kleine Hilfestellung via Fix-Beispielen zu ermöglichen.
4. Zudem sollen Content-Blocks so oft es geht wiederverwendet werden. Dazu verwendet cb eine „Out-of-the-Box“-Strategie, welche es ermöglicht, für jeden einzelnen Content-Block ein Preset und verschiedene Klassen für jedes Content-Element auszuwählen – bei Erstellung eines neuen Eintrags.

Installation:

Aktuell ist die Extension weder bei packagist noch im TYPO3-Repository gehostet, daher muss eine manuelle Installation vorgenommen werden.

1. Bei einer Composer-Umgebung genügt es den Ordner „cb-builder“ bei die anderen Extensions zu ziehen und in der composer.json den Eintrag "ds/cb-builder": "@dev" hinzuzufügen.
2. Danach lässt sich die Extension via CLI mit dem Befehl „composer update„ installieren.
3. Mit dem Befehl „cb:make“ lässt sich nun eine Extension erstellen.
   Sollte dies nicht funktionieren, bitte die Befehle „composer dump-autoload“ und danach „typo3 cache:flush“ ausführen.

Verwendung:

Erstellung eines neues Content-Blocks:

Nach dem Ausführen von dem Befehl „cb:make“ werden nun einige Informationen abgefragt.

1. Der Identifier des CB dient als einzigartiger Schlüssel zur Identifikation (unique key)
2. Anhand der ID wird nun ein Name vorgeschlagen, diesen kann man übernehmen oder selbst bestimmen.
3. Nun sieht man eine Liste aller „custom“-Extensions.
   cb_builder sollte dort auch auftauchen, da die Extension manuell installiert wurde.
   Bitte nicht cb_builder auswählen!
   Da sich die Extension noch in der internen Alpha befindet, sollte am besten eine zuvor neu erstellte Extension verwendet werden.
4. Eine einfache Beschreibung des CB.
5. Wo in dem Auswahlfenster – bei Erstellung eines neuen Content Blocks im BE – soll der neue CB angezeigt werden? Hier wird die ID des fremden CB benötigt.
6. Soll der neue CB vor oder nach dem anderen CB erscheinen?
7. Die CB-Gruppierung; bei Unsicherheit auf „default“ belassen.
8. Soll bereits ein Header- und Bodytext-Feld automatisch erstellt werden?
9. Die Dateien wurden angelegt, jedoch müssen noch 2-3 Schritte unternommen werden.
10. (Optional) WICHTIG!: Ich weiß nicht, ob es sich um einen Bug in TYPO3 handelt, wenn jedoch das Icon des CB abgeändert werden soll, so muss dies jetzt geschehen, im Nachhinein erkennt TYPO3 keine Änderung mehr an, selbst nach Löschung aller Caches. Ein eigenes Logo kann in dem Ordner „AusgewählteExtension/ContentBlocks/CB-ID/Icon“ hinzugefügt werden. Bitte die Datei „cb_icon“ löschen, da sonst eine Exception geworfen wird. Nur ein Icon in dem Ordner ist zulässig!
11. Nun liegen auch Datenbankänderung bereit, diese bitte über das BE im Modul „Maintenance->Analyze Database Structure„ anstoßen.
12. Anschließend noch über das CLI den Befehl „typo3 cache:flush“ ausführen und der neuerstellte CB sollte nun angezeigt werden.

Erstellung neuer Felder:

Über die Datei „fields.yaml“ im Ordner „Extension/ContentBlocks/Id“ lassen sich nun neue Felder definieren oder bestehende bearbeiten, bzw. löschen.

WICHTIG: Da Cross-Development getätigt wird, muss bei einer Löschung auch der Eintrag in der jeweiligen „Table.php“ (z.B. tt_content.php) unter „Extension/Configuration/TCA(/Overrides)“ gelöscht werden, da er sonst weiterhin besteht, bzw., falls „crossParsing“ aktiv ist, dieser Eintrag wieder in die fields.yaml geschrieben wird.

Die Formatierung der Felder muss mit der offiziellen Dokumentation übereinstimmen.

Beispiele gibt es in dem Ordner Examples.

Das Error-Handling unterstützt jedoch bereits in vielen Fällen bei der korrekten Programmierung.

Aktualisierung nach jeder Änderung:

Nach jeder Änderung muss der Befehl „typo3 cache:flush“ und „cb:update“ durchgeführt werden.

Ferner sollten etwaige Änderungen der Datenbankverhältnisse unter „Maintenance->Analyze Database Structure„ geprüft werden.

Presets:

Unter dem Pfad „Extension/ContentBlocks/Id/Partials“ können neue HTML-Dateien angelegt werden, welche bei dem Erstellen eines neuen CB über das BE automatisch angezeigt und ausgewählt werden können.

WICHTIG: Die Dateien unter „Extension/ContentBlocks/Id/Templates“ und „Extension/ContentBlocks/Id/Layouts“ nur bearbeiten, wenn es unbedingt notwendig ist!

CSS-Klassen:

In der Datei fields.yaml kann jedem Element das Attribut „classes“ zugewiesen werden.

Dort kann man in einer kommagetrennten Liste Klassen angeben, welche im BE bei der Erstellung eines CB ausgewählt werden können.

Die CSS-Klassen können dann unter „Extension/ContentBlocks/Id/assets“ in der jeweiligen „main.css“ oder eigenen CSS-Dateien definiert werden.

Front- und Backend bearbeiten:

Wie o.g. kann HTML-Code im Partials-Ordner und CSS im asset-Ordner modifiziert werden.

Zur automatischen Ausgabe der Klassen kann die ViewHelper-Klasse „ClassPrinter“ genutzt werden.

Beispiel:

<h2 class=“<cb:ClassPrinter value=“header“></cb:ClassPrinter>”>{data.header}</h2>

ContentBlock löschen:

Mit dem Befehl „cb:delete“ kann ein CB wieder gelöscht werden.