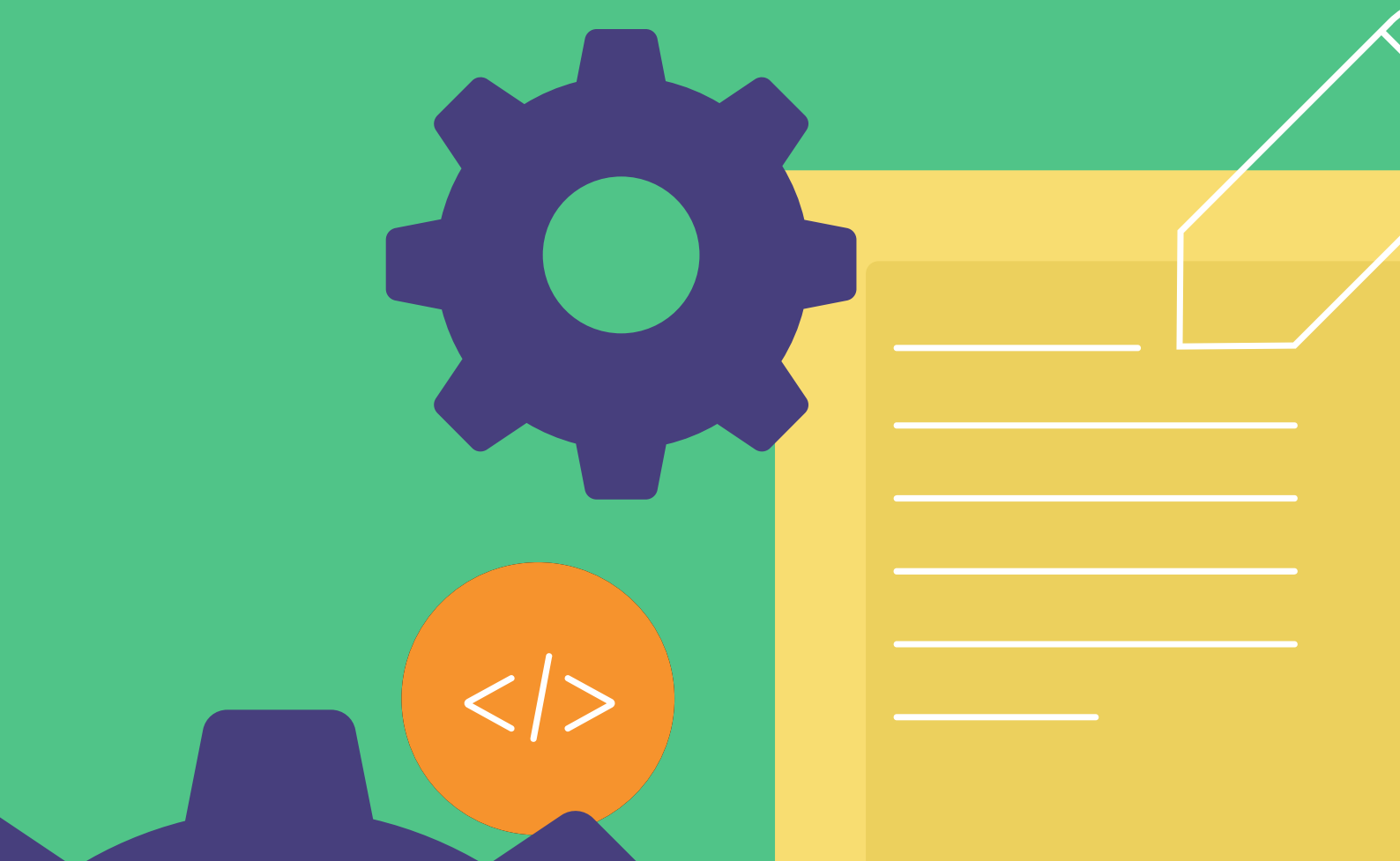
FORMCENTRIC

# Migrationsanleitung

Version 2401.6.0



# Formcentric for CoreMedia: Migrationsanleitung


Breite Str. 61, 22767 Hamburg
Deutschland



**Einschränkung der Gewährleistung**

Inhaltliche Änderungen des Handbuchs und der Software behalten wir uns ohne Ankündigung vor. Es wird keine Haftung für die Richtigkeit des Inhalts des Handbuchs oder Schäden, die sich aus dem Gebrauch der Software ergeben, übernommen.

**Warenzeichen**

Innerhalb dieses Handbuchs wird auf Warenzeichen Bezug genommen, die nicht explizit als solche ausgewiesen sind. Aus dem Fehlen einer Kennzeichnung kann nicht geschlossen werden, dass ein Name frei von Rechten Dritter ist.

Beachten Sie bitte: Ausgedruckte Exemplare unterliegen nicht dem Änderungsdienst.

# 1. Webforms auf Formcentric migrieren

Der Formcentric Editor für CoreMedia 11 ist eine eigenständige Single-Page-Anwendung, die auf dem JavaScript Framework React basiert. Diese wird aus dem CoreMedia-Studio heraus aufgerufen, wenn ein Formular editiert werden soll. Dies ermöglicht eine einfachere Konfiguration und Erweiterung des Editors, unabhängig von dem Verfahren der CoreMedia-Studio Anpassung. Allerdings müssen alle bisher vorgenommenen Erweiterungen des Webforms-Editors in das neue Format überführt werden. In dieser Anleitung wird vor allem darauf eingegangen, wie das Überführen der bestehenden Erweiterungen gelingen kann. Für die komplette Beschreibung der Anpassungsmöglichkeiten sehen sie bitte im Formcentric Entwickler Handbuch nach.

## 1.1. Basiskonfiguration des Formcentric-Formulareditors

Die Konfiguration des Editors kann in einem Settings-Dokument im Content vorgenommen werden. Eine globale Site-übergreifende Configuration kann im Pfad */Settings/Options/Settings/FormcentricSettings* hinterlegt werden. Ein Settings-Dokument mit der vollständigen Liste aller Parameter mit ihren Standardwerten ist Teil des Beispiel-Contents. Die globalen Settings können lokal Site- oder Formular-spezifisch überschrieben oder erweitert werden. Wird weder eine globale, noch eine lokale Konfiguration gefunden, startet der Editor mit einer Default-Konfiguration.

## 1.2. Hinzufügen eines zusätzlichen Formularelements

Die kundenspezifischen Formularelemente für den Formcentric-Editor werden als JSON-String konfiguriert. Fügen sie für jedes kundenspezifische Formularelement einen Eintrag in der Konfigurationsdatei *fields_custom.js* hinzu.

```
[
    {
        type: 'customerSpecificFormField',
        properties: {
        general: [
            {
            title: 'label',
            type: 'text',
        ...
```

Die Properties des bestehenden Feldes können einfach in die neue Konfiguration überführt werden. Die Konfiguration oben wurde bisher folgendermaßen umgesetzt:

```
<TextField fieldLabel="Beschriftung">
 <plugins>
  <ui:BindPropertyPlugin
```

```
   bindTo="{ValueExpressionFactory.createFromValueHolder(
   new FormChildElementValueHolder(
   config.formElementValueExpression,'label')
   )}"
   bidirectional="true"/>
</plugins>
</TextField>
```

Aus der Verwendung der Komponente *TextField* ergibt sich jetzt die Einstellung *type: 'text'*. Mit der Einstellung *title* wird konfiguriert unter welchem Schlüssel das Property in der Formulardefinition abgelegt wird. Diese Information findet sich in der Konfiguration für die ValueExpression - im o.g. Beispiel *label*. Das FieldLabel wird in der entsprechenden Sprachdatei hinterlegt. Der key wäre in diesem Fall *customerSpecificFormField.label*.

### 1.2.1. Formularelement Icon

Bisher wurde für das Icon die zu verwendende Icon-Klasse im Root-Element des Editors mit dem Attribut *icon* konfiguriert.

```
iconCls="{resourceManager.getString(
'com.monday.webforms.blueprint.icons.WebformsBlueprintIcons',
'customerSpecificFormField')}">
```

Diese Icon-Klasse kann für den neuen Editor allerdings nicht mehr weiterverwendet werden. Stattdessen muss das Icon als SVG-Datei vorliegen. Das Icon-Property wird auf den Dateinamen ohne die Endung gesetzt.

```
{
    icon: 'customField',
    ...
```

### 1.2.2. Registrieren des neuen Formularelements

Der Schritt der Registrierung des Formularelements entfällt im Formcentric-Editor da der Typ des Formularelements bei der Konfiguration mit angegeben wird. In der bisherigen Version wurde der Typ als Schlüssel verwendet, um die Klasse des Formularfeldeditors zu registrieren.

```
FormFieldsRegistry.addFormFieldClasses(
    {customerSpecificFormField:
          'com.formcentric.blueprint.studio.CustomField'} );
```

Der Typ *customerSpecificFormField* der bisher in der FormFieldRegistry verwendet wurde, wird jetzt einfach als Typ verwendet, wenn das Formularelement konfiguriert wird.

```
[
    {
        ...
        type: 'customerSpecificFormField',
        properties: {
            general: [
            ...
```

## 1.3. Implementierung einer Action

Für die Konfiguration der Kundenspezifischen Actions erweitern sie die Datei *actions_custom.js*. Bei der Migration der Eigenschaften gehen sie genauso wie bei Formularfeldern vor.

## 1.4. Anpassung der Formatvorgaben

Die Formatvorgaben werden jeweils direkt am Feld konfiguriert. Hierfür ist die Eigenschaft *format* vom Typ *dropdown_format* vorgesehen.

Das nachfolgende Beispiel zeigt die Konfiguration des E-Mail-Validators für das einzeilige Textfeld *(inputField)*.

```
{
    title: 'format',
    type: 'dropdown_format',
    properties: {
        options: {
            email: {
                enabled: true,
                fields: {
                    errormessage: {
                        title: 'errormessage',
                        type: 'text'
                                    }
                        }
                    }
                }
            }
}
```

Der angegebene Attributname (im Beispiel *email*) muss dem externen Namen des Validators entsprechen. Der Name wird auch für die Internationalisierung der

Oberfläche verwendet. In der Übersetzungsdatei wird mit der Übersetzungs-ID `<validator-name>Validator` nach einer Beschriftung für den Validator gesucht.

Sie können unter *fields* die gewünschten Felder des Validators definieren. Die verfügbaren Feldtypen finden sie im Formcentric Entwickler Handbuch.

## 1.5. Erweiterung der Formularvalidierung im Studio

Die Validierung der Formulardefinition wird wie gehabt serverseitig in der Studio-Server-App vorgenommen. Hier sind keine Anpassungen nötig.

## 1.6. Anzeige der Validierungsmeldungen

Für die Anzeige der Validierungsmeldungen ist im Formcentric-Editor keine Anpassung mehr erforderlich.