
Beratungsstellensuche

Release 6.2.0

Sebastian Schreiber

2016-10-27 14:56

CONTENTS

1	Einführung in die Erweiterung	1
1.1	Was macht die Erweiterung?	1
1.2	Wird Unterstützung benötigt?	1
2	Integrator	3
2.1	Installation	3
2.1.1	Status	3
2.1.2	Planer-Task	3
2.1.3	TypoScript	3
2.2	TypoScript	4
2.3	Templates	4
2.3.1	Templates ändern oder bearbeiten	4
	Den Pfad zu den Templates im setup anpassen	4
	Den Pfad zu den Templates über die constants anpassen	5
2.3.2	ViewHelpers	5
	MapViewHelper	5
	PaginateViewHelper	5
	RoundViewHelper	6
	DistanceViewHelper	6
	ImplodeViewHelper	6
	TranslateViewHelper	6
	TitleViewHelper	6
2.4	Praxisbeispiele	7
2.4.1	Integration mittels TypoScript	7
	Plugin mittels TypoScript einbinden	7
	Beratungsstelle in der Breadcrumb aufnehmen	7
	Title-Tag für die Detailansicht anpassen	8
	Nur mittels TypoScript	8
	ViewHelper benutzen	9
2.4.2	RealURL Konfiguration	9
	Basis Setup	9
	Erweitertes Beispiel	9
2.4.3	Linkhandler Konfiguration	11
2.4.4	Linkvalidator Konfiguration	11
2.4.5	Sitemap	11
2.4.6	Solr Konfiguration	11
3	Redakteur	13
3.1	Der Anfang	13
3.1.1	Datensätze erstellen oder importieren	13
3.1.2	Die Plugins einbinden und einrichten	13
	Detailseite	13
	Listenseite	14

	Formularansicht	14
3.2	Datensätze	14
3.2.1	Beratungsstelle	14
	Felder	14
3.2.2	Beratungsart	14
	Felder	14
4	Entwickler	15
4.1	Flexforms erweitern	15
4.1.1	Formularfelder ergänzen	15
4.1.2	Layoutauswahl ergänzen	15
4.2	Signals & Hooks	15
4.2.1	Signals	15
	Beispiel	16
4.2.2	Hooks	16
	Domain/Repository/EntryRepository.php findDemanded	17
	Beispiel	17
4.3	Erweiterung des Model	17
4.3.1	1) Neue Felder hinzufügen	18
	SQL	18
	TCA	18
4.3.2	2) Klasse registrieren	19
	Eigene Klasse	19
	Clear system cache	19
	XML-Importer	20

EINFÜHRUNG IN DIE ERWEITERUNG

1.1 Was macht die Erweiterung?

Die Erweiterung ermöglicht die Pflege und Darstellung von Beratungsstellen (Locations) auf Basis des CMS TYPO3.

Durch die Verwendung von zahlreichen Hooks, Signals und TypoScript-Settings kann die Erweiterung flexibel an die eigenen Anforderungen und Wünsche angepasst werden.

Über einen bereitgestellten Importer-Task können Beratungsstellen aus dem System von <http://www.bzga-rat.de/adm> über die vorhandene XML-Schnittstelle importiert werden.

1.2 Wird Unterstützung benötigt?

Sollten Sie bei der Installation, Konfiguration oder Weiterentwicklung Unterstützung benötigen, dann können Sie sich jederzeit...

INTEGRATOR

2.1 Installation

Diese Erweiterung wird wie jede andere Erweiterung des TYPO3 CMS installiert:

1. Wechseln Sie zum Modul **Extension Manager**.
2. Installieren Sie die Erweiterung über den Upload-Button im Extension Manager.
3. Führen Sie nach der Installation das mitgelieferte Update-Skript der Erweiterung innerhalb des Extension Manager aus.

2.1.1 Status

1. Wechseln Sie zum Modul **Berichte**. Wählen Sie im oberen Select-Menü den Punkt **Status Report** aus. Hier sollte unter dem key *bzgaberatungsstellensuche* alles grün sein. Dies ist wichtig, damit externe Services wie der XML-Importer oder die GoogleMaps-API angesprochen werden können.

2.1.2 Planer-Task

Möchten Sie die XML-Schnittstelle von bzga-rat.de nutzen, dann legen Sie im Modul **Planer** einen neuen **Task** an.

1. Wechseln Sie zum Modul **Planer**.
2. Legen Sie einen neuen **Task** mit folgenden Einstellungen an:
 - Klasse: Extbase-CommandController-Task
 - Typ: wiederkehrend
 - Start: 13:32 26-10-2016 (z.B.)
 - Ende: 13:32 26-10-2099 (z.B.)
 - Häufigkeit: */15 * * * * (alle 15 Minuten, z.B.)
 - CommandController Command: BzgaBeratungsstellensuche Import: ImportFromUrl
3. Als Url geben Sie bitte die Ihnen vorliegende Url zur XML-Datei an und im Feld **pid** tragen Sie den Ordner zum Abspeichern der Beratungsstellen ein.

2.1.3 TypeScript

Für die korrekte Verwendung der Erweiterung müssen Sie zunächst das mitgelieferte Basis TypeScript inkludieren.

1. Wechseln Sie zur **Root-Seite** Ihrer Installation.
2. Wechseln Sie zum Modul **Template** und wählen Sie im Auswahl-Menü *Info/Modify* aus.

3. Klicken Sie den Link **Edit the whole template record** und wechseln Sie zum Reiter *Enthält*.
4. Wählen Sie **Beratungsstellensuche (bzga_beratungsstellensuche)** im Feld *Statische Templates einschließen (aus Erweiterungen)*: aus.

2.2 TypeScript

Die Erweiterung enthält einige TypeScript-Eigenschaften (settings), die zur individuellen Konfiguration der Erweiterung dienen. Im Folgenden werden die einzelnen Eigenschaften beschrieben.

2.3 Templates

Dieses Kapitel beschreibt wie Sie die mitgelieferten Templates überschreiben und die mitgelieferten ViewHelpers einsetzen können.

Hint: Das Markup der Templates basiert auf Twitter bootstrap (<http://getbootstrap.com/>)

2.3.1 Templates ändern oder bearbeiten

Warning: Die Erweiterung basiert auf der Template Engine **fluid**. Grundlegende Kenntnisse in Aufbau und Verwendung von fluid werden vorausgesetzt.

Den Pfad zu den Templates im setup anpassen

Sie sollten niemals die Original-Templates der Erweiterung verändern. Diese werden bei einem Update ansonsten überschrieben.

Wie bei allen Extensions die auf extbase basieren, befinden sich die Template im Ordner `Resources/Private/`.

Sollten Sie ein Template anpassen, kopieren Sie sich das entsprechende Template an die gewünschte Stelle. Die Templates können dabei in einer eigenen Erweiterung liegen oder einfach im fileadmin (nicht zu empfehlen).

Der Einfachheit halber gehen wir im folgenden Beispiel davon aus, dass die Templates im fileadmin-Ordner abgelegt werden.

Der folgende Teil würde im **setup** Bereich Ihres TypoScripts stehen:

```
plugin.tx_news {
    view {
        templateRootPaths >
        templateRootPaths {
            0 = EXT:bzga_beratungsstellensuche/Resources/Private/
↪Templates/
            1 = fileadmin/templates/ext/bzga_beratungsstellensuche/
↪Templates/
        }
        partialRootPaths >
        partialRootPaths {
            0 = EXT:bzga_beratungsstellensuche/Resources/Private/
↪Partials/
            1 = fileadmin/templates/ext/bzga_beratungsstellensuche/
↪Partials/
        }
        layoutRootPaths >
```



```

        layoutRootPaths {
            0 = EXT:bzga_beratungsstellensuche/Resources/Private/
→Layouts/
            1 = fileadmin/templates/ext/bzga_beratungsstellensuche/
→Layouts/
        }
    }
}

```

Mit diesem TypoScript wird zunächst im fileadmin nachgesehen ob ein angefordertes Template, Partial oder Layout existiert. Sollte dies nicht der Fall sein, kommt es zu einem Fallback auf das Original.

Hint: Bitte achten Sie auf die Plural-Schreibweise. Also z.B nicht `templateRootPath` sondern `templateRootPaths`.

Den Pfad zu den Templates über die constants anpassen

Es gibt auch die Möglichkeit die Pfade über die **constants** zu setzen.

```

plugin.tx_news {
    view {
        templateRootPath = fileadmin/templates/ext/news/Templates/
        partialRootPath = fileadmin/templates/ext/news/Partials/
        layoutRootPath = fileadmin/templates/ext/news/Layouts/
    }
}

```

2.3.2 ViewHelpers

Die Erweiterung liefert einige nützliche **ViewHelpers** mit, die entweder bereits in den bestehenden Templates verwendet werden oder von Ihnen in Ihren Anpassungen genutzt werden könnten. Es sind hier nicht alle entwickelten ViewHelper aufgeführt, da einige lediglich für "interne Zwecke" genutzt werden.

Um die ViewHelper zu nutzen müssen Sie zunächst in den jeweiligen fluid-Dateien den namespace über folgende Anweisung importieren:

```
{namespace bzga=Bzga\BzgaBeratungsstellensuche\ViewHelpers}
```

MapViewHelper

Der MapViewHelper generiert eine Karte auf Basis von GoogleMaps. Die Darstellung der Karte kann über Hooks und über individuell TS-Settings angepasst werden.

Klasse: `Classes/ViewHelpers/Widget/MapViewHelper.php`

```
<bzga:widget.map demand="{demand}" entry="{entry}" settings="{settings}"/>
```

PaginateViewHelper

Für den Seitenbrowser wurde der in fluid enthaltene PaginateViewHelper erweitert. Bitte also in allen Fällen im Kontext der Beratungsstellen diesen ViewHelper nutzen.

Klasse: `Classes/ViewHelpers/Widget/PaginateViewHelper.php`

```
<bzga:widget.paginate as="paginatedEntries" demand="{demand}" objects="{entries}">
  <f:for each="{paginatedEntries}" as="entry">
    {entry}
  </f:for>
</bzga:widget.paginate>
```

RoundViewHelper

Der RoundViewHelper findet aktuell gemeinsam mit dem DistanceViewHelper Verwendung und rundet ein Float-Wert nach einem definierten Präzisionswert.

Klasse: Classes/ViewHelpers/Math/RoundViewHelper.php

```
<bzga:math.round precision="2">{float}</bzga:math.round>
```

DistanceViewHelper

Der DistanceViewHelper berechnet die Entfernung von zwei Objekten vom Typ GeopositionInterface.

Klasse: Classes/ViewHelpers/DistanceViewHelper.php

```
<bzga:distance demandPosition="{demand}" location="{entry}" />
```

ImplodeViewHelper

Der Implode-ViewHelper kann dazu verwendet werden, aus einem Array oder Traversable einen string zu generieren. Die Objekte innerhalb des Arrays oder Traversable müssen die __toString Methode implementieren.

Klasse: Classes/ViewHelpers/ImplodeViewHelper.php

```
<bzga:implode glue=",">{array}</bzga:implode>
```

TranslateViewHelper

Der TranslateViewHelper ist eine Erweiterung des fluid eigenen TranslateViewHelper und dient dazu die Lokalisierungslabels aus allen “registrierten Erweiterungen” automatisch zu ziehen. Eine Erweiterung kann mittels des folgenden Aufrufs in der Datei ext_localconf.php registriert werden:

```
<?php
\BZgA\BzgaBeratungsstellensuche\Utility\ExtensionManagementUtility::
    registerExtensionKey($_EXTKEY, $priority);
```

Klasse: Classes/ViewHelpers/TranslateViewHelper.php

Bitte verwenden Sie in allen Dateien im Kontext der Beratungsstellensuche diesen TranslateViewHelper.

```
<bzga:translate key="telephone" />
```

TitleViewHelper

Der TitleViewHelper ermöglicht das Überschreiben des Title-Tag auf der Webseite.

Klasse: Classes/ViewHelpers/TitleViewHelper.php

```
<bzga:title>My new title</bzga:title>
```

2.4 Praxisbeispiele

2.4.1 Integration mittels TypeScript

Diese Seite gibt Ihnen einige Beispiele wie Sie die Erweiterung bzga_beratungsstellensuche mittels TypeScript in Ihre Seite integrieren können.

Plugin mittels TypeScript einbinden

Möchten Sie zum Beispiel die Formularansicht der Beratungsstellensuche auf jeder Seite Ihres Auftritts anzeigen, dann benutzen Sie folgendes TypeScript:

```
lib.beratungsstellensuche = USER
lib.beratungsstellensuche {
    userFunc = TYPO3\CMS\Extbase\Core\Bootstrap->run
    extensionName = BzgaBeratungsstellensuche
    pluginName = Pil
    vendorName = BZgA

    switchableControllerActions {
        Entry {
            1 = form
        }
    }

    settings < plugin.tx_bzgaberatungsstellensuche.settings
    settings {
        formFields = location,kilometers
        listPid = 12345
    }
}

[globalVar = GP:tx_bzgaberatungsstellensuche_pil|__trustedProperties = /\w+/ ]
lib.beratungsstellensuche = USER_INT
[global]
```

Jetzt können Sie das Objekt lib.beratungsstellensuche an gewünschter Stelle einbinden.

Beratungsstelle in der Breadcrumb aufnehmen

Möchten Sie die Beratungsstelle in Ihrer Breadcrumb mit aufnehmen, benutzen Sie den folgenden TypeScript-Code:

```
lib.breadcrumb = COA
lib.breadcrumb {
    stdWrap.wrap = <ul class="breadcrumb">|</ul>

    10 = HMENU
    10 {
        special = rootline
        #special.range = 1

        1 = TMENU
        1 {
            noBlur = 1
        }
    }
}
```

```

        NO = 1
        NO {
            wrapItemAndSub = <li>|</li>
            ATagTitle.field = subtitle // title
            stdWrap.htmlSpecialChars = 1
        }

        CUR <.NO
        CUR {
            wrapItemAndSub = <li class="active">|</li>
            doNotLinkIt = 1
        }
    }

    # Add news title if on single view
    20 = RECORDS
    20 {
        stdWrap.if.isTrue.data = GP:tx_bzgaberatungsstellensuche_pil|entry
        dontCheckPid = 1
        tables = tx_bzgaberatungsstellensuche_domain_model_entry
        source.data = GP:tx_bzgaberatungsstellensuche_pil|entry
        source.intval = 1
        conf.tx_bzgaberatungsstellensuche_domain_model_entry = TEXT
        conf.tx_bzgaberatungsstellensuche_domain_model_entry {
            field = title
            htmlSpecialChars = 1
        }
        stdWrap.wrap = <li>|</li>
        stdWrap.required = 1
    }
}

```

Title-Tag für die Detailansicht anpassen

Möchten Sie in der Detailansicht einer Beratungsstelle das Title-Tag anpassen, können Sie folgende 2 Wege gehen:

Nur mittels TypoScript

```

[globalVar = GP:tx_bzgaberatungsstellensuche_pil|entry > 0]

config.noPageTitle = 2

temp.title = RECORDS
temp.title {
    dontCheckPid = 1
    tables = tx_bzgaberatungsstellensuche_domain_model_entry
    source.data = GP:tx_bzgaberatungsstellensuche_pil|news
    source.intval = 1
    conf.tx_bzgaberatungsstellensuche_domain_model_entry = TEXT
    conf.tx_bzgaberatungsstellensuche_domain_model_entry {
        field = title
        htmlSpecialChars = 1
    }
    wrap = <title>|</title>
}

page.headerData.1 >
page.headerData.1 < temp.newsTitle

```

```
[global]
```

ViewHelper benutzen

```
<bzga:title>{entry.title}</bzga:title>
```

2.4.2 RealURL Konfiguration

Um die Urls mittels der Erweiterung **RealURL** ansprechender zu gestalten können Sie die im Folgenden aufgeführten Konfigurationen in Ihrem Projekt integrieren.

Basis Setup

Die einfachste Art und Weise besteht darin, den folgenden Code in den *postVarSets/_DEFAULT* Abschnitt zu kopieren:

```
// EXT:bzga_beratungsstelle start
'beratungsstelle' => array(
    array(
        'GETvar' => 'tx_bzgaberatungsstellensuche_pil[action]',
    ),
    array(
        'GETvar' => 'tx_bzgaberatungsstellensuche_pil[controller]',
    ),
    array(
        'GETvar' => 'tx_bzgaberatungsstellensuche_pil[entry]',
        'lookUpTable' => array(
            'table' => 'tx_bzgaberatungsstellensuche_domain_model_entry
→ ',
            'id_field' => 'uid',
            'alias_field' => 'title',
            'addWhereClause' => ' AND NOT deleted',
            'useUniqueCache' => 1,
            'useUniqueCache_conf' => array(
                'strtolower' => 1,
                'spaceCharacter' => '-',
            ),
            'languageGetVar' => 'L',
            'languageExceptionUids' => '',
            'languageField' => 'sys_language_uid',
            'transOrigPointerField' => 'l10n_parent',
            'autoUpdate' => 1,
            'expireDays' => 180,
        ),
    ),
),
// EXT:bzga_beratungsstelle end
```

Erweitertes Beispiel

Eine weitere Möglichkeit besteht darin, dass Sie die Konfiguration in den Abschnitt *fixedPostVars/_DEFAULT* anlegen:

```
<?php
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['realurl'][_DEFAULT] = array(
    'fixedPostVars' => array(
        'entryDetailConfiguration' => array(
            array(
                'GETvar' => 'tx_bzgageberungsstellensuche_
↪pil[action]',
                'valueMap' => array(
                    'show' => '',
                ),
                'noMatch' => 'bypass'
            ),
            array(
                'GETvar' => 'tx_bzgageberungsstellensuche_
↪pil[controller]',
                'valueMap' => array(
                    'Entry' => '',
                ),
                'noMatch' => 'bypass'
            ),
            array(
                'GETvar' => 'tx_bzgageberungsstellensuche_
↪pil[entry]',
                'lookUpTable' => array(
                    'table' => 'tx_bzgageberungsstellensuche_
↪domain_model_entry',
                    'id_field' => 'uid',
                    'alias_field' => 'title',
                    'addWhereClause' => ' AND NOT deleted',
                    'useUniqueCache' => 1,
                    'useUniqueCache_conf' => array(
                        'strtolower' => 1,
                        'spaceCharacter' => '-'
                    ),
                    'languageGetVar' => 'L',
                    'languageExceptionUids' => '',
                    'languageField' => 'sys_language_uid',
                    'transOrigPointerField' => 'l10n_parent',
                    'autoUpdate' => 1,
                    'expireDays' => 180,
                )
            )
        ),
        '70' => 'entryDetailConfiguration',
    ),
    'postVarSets' => array(
        '_DEFAULT' => array(
            'controller' => array(
                array(
                    'GETvar' => 'tx_bzgageberungsstellensuche_
↪pil[action]',
                    'noMatch' => 'bypass'
                ),
                array(
                    'GETvar' => 'tx_bzgageberungsstellensuche_
↪pil[controller]',
                    'noMatch' => 'bypass'
                )
            )
        ),
    ),
),
```

```
) ;
```

2.4.3 Linkhandler Konfiguration

Sollte die Extension **linkhandler** in Ihrer Installation verwendet werden, wird automatisch von der Erweiterung bzga_beratungsstellensuche ein Basis-Setup in der TSConfig inkludiert.

Die Extension **linkhandler** ermöglicht die Verlinkung von beliebigen Datensätzen innerhalb von Textfeldern. Nähere Information entnehmen Sie bitte hier: <https://github.com/Intera/typo3-extension-linkhandler>

Dieses Setup befindet sich in der Datei: EXT:bzga_beratungsstellensuche/Configuration/TsConfig/Page/mod

Damit die Verlinkungen auch im Frontend wirksam werden, muss ein weiteres **statisches TypoScript-Template** in Ihrer Root-Seite inkludiert werden. Dieses Template heißt: **Beratungsstellensuche - Linkhandler**

2.4.4 Linkvalidator Konfiguration

Sollte die Extension **linkvalidator** in Ihrer Installation verwendet werden, wird automatisch von der Erweiterung bzga_beratungsstellensuche ein Basis-Setup in der TSConfig inkludiert.

Die Core-Extension **linkvalidator** ermöglicht die Überprüfung von Verlinkungen innerhalb konfigurierter Tabellen und Felder. Nähere Information entnehmen Sie bitte hier: <https://docs.typo3.org/typo3cms/extensions/linkvalidator/Introduction/Index.html>

Dieses Setup befindet sich in der Datei: EXT:bzga_beratungsstellensuche/Configuration/TsConfig/Page/mod

2.4.5 Sitemap

Sollte die Extension **dd_googlesitemap** in Ihrer Installation verwendet werden, dann können Sie eine **spezielle Sitemap** für die Beratungsstellen generieren lassen.

Der **Aufruf der Sitemap** könnte dabei folgendermaßen aussehen: http://www.ihre-domain.com?eID=dd_googlesitemap&sitemap=bzga_beratungsstellensuche&singlePid=SINGLE_PID&pidList=STORAGE_FOLDER_PID

Hint: Die Parameter SINGLE_PID und STORAGE_FOLDER_PID müssen bitte durch die korrekten Angaben ersetzt werden.

2.4.6 Solr Konfiguration

Sollte die Extension **solr** in Ihrer Installation verwendet werden, können Sie ein weiteres **statisches TypoScript-Template** auf Ihrer Root-Seite inkludieren.

Dieses Template heißt: **Beratungsstellensuche - Solr**

Das Zusammenspiel mit solr beim XML-Import funktioniert einwandfrei, da beim Import-Vorgang die DataHandler-API genutzt wird und somit solr bei Einfügen/Aktualisieren oder Löschen einer Beratungsstelle darüber in Kenntnis gesetzt wird.

Note: Sollten Sie neben solr auch die Erweiterung **solrgeo** nutzen, müssen Sie die Integration mit der Erweiterung **bzga_beratungsstellensuche** selbständig vornehmen. Wir freuen uns über ein Feedback bei erfolgreicher Integration.

REDAKTEUR

3.1 Der Anfang

- *Datensätze erstellen oder importieren*
- *Die Plugins einbinden und einrichten*

3.1.1 Datensätze erstellen oder importieren

Um überhaupt Beratungsstellen im Frontend darstellen zu können, müssen diese zunächst im Backend vorhanden sein.

1. Erstellen Sie einen neuen Seitenordner oder verwenden Sie einen bestehenden.
2. Sie können jetzt entweder neue Datensätze auf die bekannte Art und Weise im Backend manuell anlegen oder Sie nutzen die bereitgestellte XML-Schnittstelle zum Import der Beratungsstellen über die Website <http://www.bzga-rat.de/adm>.
3. Für die Verwendung der XML-Schnittstelle müssen Sie zunächst einen Task im Planer anlegen. Sehe hierzu auch: *Planer-Task*.

3.1.2 Die Plugins einbinden und einrichten

Für die Suche und Darstellung der Listen- und Detailansicht der Beratungsstellen müssen Sie das entsprechende Plugin auf den jeweiligen Seiten hinterlegen.

Detailseite

1. Legen Sie eine neue Seite "Detail" (oder einem frei wählbaren anderen Namen) im Seitenbaum an. Die Seite sollte als im Menü verborgen angelegt werden. Auf dieser Seite hinterlegen Sie das Plugin "Beratungsstellen". Unter dem "Reiter" Einstellungen wählen Sie im Feld Ansicht "Detailansicht" aus.
2. Unter dem Reiter "Erweiterte Einstellungen" können Sie optional eine Seite für zurück und eine Seite für die Detailansicht anderer Beratungsstellen hinterlegen. Diese beiden Einstellungen können aber auch global über die TypoScript Eigenschaften definiert werden und müssen nicht im Plugin selbst hinterlegt werden. Siehe hierzu:
3. Unter dem Reiter "Template" kann optional ein hinterlegtes Layout ausgewählt werden. Nähere Informationen hierzu befinden sich im Bereich: *Layoutauswahl ergänzen*.
4. Speichern Sie nach erfolgreicher Konfiguration das Plugin-Inhaltselement ab.

Listenansicht

1. Legen Sie eine neue Seite mit der Bezeichnung “Liste” (oder einem frei wählbaren anderen Namen) im Seitenbaum an und hinterlegen Sie dort ebenfalls ein Plugin vom Typ “Beratungsstellen”.
2. Die bereits vorausgewählte Ansicht “Listenansicht” kann bestehen bleiben.
3. Sie können hier optional im Feld “Startingpoint” den Ordner angeben in dem die Beratungsstellen abgespeichert wurden. Dies kann aber auch global über die TypeScript-Eigenschaften definiert werden.
4. Unter dem Reiter “Erweiterte Einstellungen” können Sie zusätzliche einige lokale Angaben im Plugin vornehmen. Diese Angaben können aber auch global über die TypeScript-Eigenschaften gesetzt werden.
5. Speichern Sie nach erfolgreicher Konfiguration das Plugin-Inhaltselement ab.

Formularansicht

Sollten Sie zusätzlich zur Listen- und Detailansicht eine vorgeschaltete Formularansichtsseite benötigen, dann führen Sie bitte folgende Schritte aus:

1. Legen Sie eine neue Seite mit der Bezeichnung “Formular” (oder einem frei wählbaren anderen Namen) im Seitenbaum an und hinterlegen Sie dort ein Plugin vom Typ “Beratungsstellen”.
2. Unter dem “Reiter” Einstellungen wählen Sie im Feld Ansicht “Formular” aus. Alle möglichen Konfigurationsoptionen sollten bereits aus den vorher beschriebenen Ansichten bekannt sein.

Hint: Es wird vermutlich gewünscht sein, dass sich auf allen Seiten Ihres Webauftrittes ein Suchformular für die Beratungsstellen befindet. Für eine mögliche Lösung dieser Anforderung schauen Sie sich bitte den Bereich *Plugin mittels TypeScript einbinden* an.

3.2 Datensätze

EXT:bzga_beratungsstellensuche bringt von Hause aus zwei unterschiedliche Datensatztypen mit, die hier näher beschrieben werden:

3.2.1 Beratungsstelle

Der Datensatz Beratungsstelle ist der zentrale Datensatztyp dieser Erweiterung.

Felder

3.2.2 Beratungsart

Die Beratungsarten sind nicht obligatorisch und sind lediglich eine Möglichkeit die Beratungsstellen zu gruppieren und im Frontend filtern zu können.

Felder

ENTWICKLER

4.1 Flexforms erweitern

Einige Felder der Flexform-Konfiguration können mit Hilfe von Hooks erweitert werden.

4.1.1 Formularfelder ergänzen

Dieses Feld ist dazu da, damit der Redakteur bei der Konfiguration die Möglichkeit hat, die Anzeige der Formularfelder gezielt und individuell steuern zu können. Als Entwickler können wir hier, wenn alle Vorbereitungen wie Template und Datenbanklogik dafür vorbereitet wurden, neue Felder für die Auswahl ergänzen. Dafür muss in der `ext_localconf.php` oder `LocalConfiguration.php` folgender Aufruf erstellt werden:

```
<?php
\BZgA\BzgaBeratungsstellensuche\Utility\ExtensionManagementUtility::
    ↪addAdditionalFormField(array('LLL:EXT:your_extension_key/path_to_locales_file:
    ↪label', 'fieldname'));
```

4.1.2 Layoutauswahl ergänzen

Sollte es für die Frontenddarstellung unterschiedliche Layoutausgaben geben, kann dies über die Bereitstellung von Layouttemplates ermöglicht werden. Um weitere Layouts bereitzustellen, muss folgende Angabe in der `ext_localconf.php` oder `LocalConfiguration.php` gemacht werden:

```
<?php
$GLOBALS['TYPO3_CONF_VARS']['EXT']['bzga_beratungsstellensuche']['templateLayouts
    ↪']['myext'] = array('My Title', 'my value');
```

Im Template erfolgt der Zugriff auf die Angabe folgendermaßen: `{settings.templateLayout}` und kann beispielsweise in einer Fallunterscheidung (Condition) verwendet werden.

4.2 Signals & Hooks

Einige bereitgestellte Hooks und Signals können zur Anpassung der Erweiterung genutzt werden. Nach Bedarf können gerne weitere Hooks und Signals bereitgestellt werden. Setzen Sie sich diesbezüglich gerne mit uns in Verbindung: *Wird Unterstützung benötigt?*.

4.2.1 Signals

Alle bereitgestellten Signals können in der Datei `Classes/Events.php` der Erweiterung `bzga_beratungsstellensuche` eingesehen werden und sind ausreichend kommentiert.

Beispiel

Als Beispiel möchten wir für die Formularansicht eine weitere Variable der View hinzufügen.

```
<?php
// Dies ist der Code-Abschnitt in der Datei Classes/Controller/EntryController.php
$assignedViewValues = array(
    'demand' => $demand,
    'kilometers' => $kilometers,
    'categories' => $categories,
    'countryZonesGermany' => $countryZonesGermany
);
$this->emitActionSignal(Events::FORM_ACTION_SIGNAL, $assignedViewValues)
```

Um dieses Signal zu nutzen, erstellen Sie einen Slot in Ihrer eigenen Erweiterung. Dafür wird ein Eintrag in Ihrer `ext_localconf.php` benötigt:

```
<?php
/** @var \TYPO3\CMS\Extbase\SignalSlot\Dispatcher $signalSlotDispatcher */
$signalSlotDispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::
    ↪makeInstance(\TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class);

// Extend form view
$signalSlotDispatcher->connect(
    \BZgA\BzgaBeratungsstellensuche\Controller\EntryController::class,
    \BZgA\BzgaBeratungsstellensuche\Events::FORM_ACTION_SIGNAL,
    \YOUR_VENDOR\YOUR_EXTKEY\Slots\EntryController::class,
    'formAction'
);
```

Ihr Slot könnte dann wie folgt aussehen:

```
<?php

namespace YOUR_VENDOR\YOUR_EXTKEY\Slots;
class EntryController
{
    public function listAction($variables)
    {
        ↪$variables = array_merge($variables, array('myVariable' => 'myValue'));
        ↪return array(
            ↪'extendedVariables' => $variables,
        );
    }
}
```

Hint: Schauen Sie sich bitte die Datei `Classes/Events.php` in Ruhe an. Sie werden darüber schnell erfahren, welche Teile in der Erweiterung flexibel erweitert werden können.

4.2.2 Hooks

Es gibt aktuell verschiedenen Stellen an denen statt Signals Hooks zum Einsatz kommen. Für die Erweiterung der Flexforms haben wir bereits zwei Hooks im Einsatz gesehen. Siehe hierzu: [Flexforms erweitern](#).

Für das Frontend gibt es weitere interessante Hooks.

Domain/Repository/EntryRepository.php findDemedanded

Dieser Hook ermöglicht die Abfrage zur Filterung der Einträge um seine eigene Logik zu erweitern.

Beispiel

Dieses Beispiel ergänzt die Abfrage um eine weitere Bedingung, so dass nur Einträge angezeigt werden, die das Keyword “unabhängig” enthalten.

Als Erstes muss der Hook in der Datei `ext_localconf.php` erstellt werden:

```
<?php

$GLOBALS['TYPO3_CONF_VARS']['EXT']['bzga_beratungsstellensuche']['Domain/Repository/
→EntryRepository.php']['findDemedanded'][$_EXTKEY]
    = 'YOUR_VENDOR\YOUR_EXTKEY\Hooks\Repository->modify';
```

Jetzt erstellen Sie die Datei `Classes/Hooks/Repository.php`:

```
<?php

namespace YOUR_VENDOR\YOUR_EXTKEY\Hooks;

use Bzga\BzgaBeratungsstellensuche\Domain\Model\Dto\Demand;
use TYPO3\CMS\Extbase\Persistence\QueryInterface;

class Repository
{
    public function modify(array $params)
    {
        $query = $params['query'];
        /* @var $query QueryInterface */
        $constraints = &$params['constraints'];
        /* @var $constraints array */
        $demand = $params['demand'];
        /* @var $demand Demand */
        $constraints[] = $query->like('keywords', '%unabhängig%');
    }
}
```

Hint: Um die bereitgestellten Hooks ausfindig zu machen, können Sie bspw. einmal folgenden Befehl ausführen:
`grep -n -C 5 "[EXT]"[bzga_beratungsstellensuche]" -r bzga_beratungsstellensuche/Classes/`

Hint: Bitte passen Sie den Vendor-Präfix und den Extension-Key an ihre Gegebenheiten an.

4.3 Erweiterung des Model

In diesem Kapitel lernen Sie, wie Sie im Model und der Datenbank neue Felder zu bestehenden Models hinzufügen können.

Die Erweiterung ermöglicht die dynamische Überschreibung der Basismodels Entry, Category und Demand ohne dass diese über Vererbung erweitert werden. Dafür wird eine Proxy-Klasse unter Verwendung des TYPO3 Caching Frameworks im Ordner `typo3temp/Cache/Code/bzga_beratungsstellensuche` abgespeichert. Es ist wichtig zu wissen, dass dieses Konzept existiert.

Warning: Dieses Konzept hat folgende Nachteile:

- Benutzen Sie keine use statements am Anfang der Datei. Diese werden ignoriert!
- Es ist nicht möglich eine bestehende Methode zu überschreiben.

4.3.1 1) Neue Felder hinzufügen

Der Extension-Key in diesem Beispiel lautet: bzga_beratungsstellensuche_extended.

Note: Wir gehen in diesem Beispiel davon aus, dass Sie die grundlegenden Kenntnisse zur Erstellung einer TYPO3-Erweiterung besitzen.

Es werden grundsätzlich 2 Dateien benötigt.

SQL

Erstellen Sie die Datei `ext_tables.sql` im Root-Verzeichnis Ihrer Erweiterung mit folgendem Inhalt:

```
#
# Table structure for table 'tx_bzgaberatungsstellensuche_domain_model_entry'
#
CREATE TABLE tx_bzgaberatungsstellensuche_domain_model_entry (
    alternative_email varchar(255) DEFAULT '' NOT NULL,
);
```

TCA

Erstellen Sie eine Datei namens `tx_bzgaberatungsstellensuche_domain_model_entry.php` im Ordner `Configuration/TCA/Overrides/`.

```
<?php
defined('TYPO3_MODE') or die();

$fields = array(
    'alternative_email' => array(
        'exclude' => 1,
        'label' => 'Alternative E-Mail Adresse',
        'config' => array(
            'type' => 'input',
            'size' => 30,
            'eval' => 'trim',
            'wizards' => array(
                '_PADDING' => 2,
                'link' => array(
                    'type' => 'popup',
                    'title' => 'LLL:EXT:cms/locallang_ttc.xlf:
→header_link_formlabel',
                    'icon' => 'link_popup.gif',
                    'module' => array(
                        'name' => 'wizard_element_browser',
                        'urlParameters' => array(
                            'mode' => 'wizard'
                        )
                    ),
                    'JSOpenParams' => 'height=600,width=800,
→status=0,menubar=0,scrollbars=1'
```

```

        ),
    ),
);

\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addTCAcolumns('tx_
→bzgaberatungsstellensuche_domain_model_entry', $fields);
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addToAllTCAtypes('tx_
→bzgaberatungsstellensuche_domain_model_entry', 'alternative_email');
```

4.3.2 2) Klasse registrieren

Damit die Erweiterung bzga_beratungsstellensuche Kenntnis über Ihre Erweiterung hat, erstellen Sie bitte im Ordner Configuration/DomainModelExtension eine Datei namens BzgaBeratungsstellensuche.txt mit folgendem Inhalt:

```
Domain/Model/Entry
```

Eigene Klasse

Erstellen Sie nun eine Datei typo3conf/ext/bzga_beratungsstellensuche_extended/Classes/Domain/Model/Entry.php

```
<?php

namespace BZgA\BzgaBeratungsstellensucheExtended\Domain\Model;

/**
 * Entry
 */
class Entry extends \BZgA\BzgaBeratungsstellensuche\Domain\Model\Entry {

    /**
     * @var string
     */
    protected $alternativeEmail;

    /**
     * @return string
     */
    public function getAlternativeEmail()
    {
        return $this->alternativeEmail;
    }

    /**
     * @param string $alternativeEmail
     */
    public function setAlternativeEmail($alternativeEmail)
    {
        $this->alternativeEmail = $alternativeEmail;
    }
}
```

Clear system cache

Jetzt müssen Sie nur noch den **system cache** leeren. Danach können Sie überall auf dieses Feld zurückgreifen.

Hint: In den meisten Fällen wird die Anforderung doch etwas komplexer sein. Sollten Sie nicht weiterkommen, stellen wir Ihnen gerne eine Erweiterung zu Anschauungszwecken zur Verfügung.

XML-Importer

Möchten Sie das neue Feld aus dem obigen Beispiel aus dem XML auslesen und in der Datenbank abspeichern, müssen Sie in unserem Beispiel noch einen Slot erstellen.

Hint: Zum Thema Slots schauen Sie auch hier: [Signals](#).

Zunächst registrieren Sie den Slot für das entsprechende Signal in der `ext_localconf.php`

```
<?php

// Extend name converter
$signalSlotDispatcher->connect (

    ↳ \BZgA\BzgaBeratungsstellensuche\Domain\Serializer\NameConverter\EntryNameConverter:
    ↳ :class,
        \BZgA\BzgaBeratungsstellensuche\Events::SIGNAL_MapNames,
        \YOUR_VENDOR\YOUR_EXTKEY\Slots\EntryNameConverter::class,
        'mapNames'
);
```

Jetzt müssen Sie noch die dazugehörige Slot-Klasse schreiben:

```
<?php
namespace YOUR_VENDOR\YOUR_EXTKEY\Slots;

class EntryNameConverter
{
    /**
     * @param array $mapNames
     * @return array
     */
    public function mapNames(array $mapNames = array())
    {
        $mapNames = array_merge($mapNames, array(
            'alt_email' => 'alternative_email',
        ));

        return array(
            'extendedMapNames' => $mapNames,
        );
    }
}
```

Hint: In diesem Beispiel enthält das XML im Knoten **entry** einen weiteren Unterknoten mit der Bezeichnung **alt_email** und wird dem neuen Feld **alternative_email** zugeordnet. Der restliche Ablauf ist Magie.
