bzga_beratungsstellensuche Documentation

Release stable

19.01.2022

Inhaltsverzeichnis

KAPITEL 1

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-Schnitstelle importiert werden.

1.2 Wird Unterstützung benötigt?

Sollten Sie bei der Installation, Konfiguration oder Weiterentwicklung Unterstützung benötigen, dann können Sie sich jederzeit an die Autoren der Erweiterung wenden. Wir unterstützen Sie im Rahmen Ihrer Projekte mit Rat und Tat.

KAPITEL 2

Integrator

2.1 Installation

Diese Erweiteurng 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-Schittstelle 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 TypoScript

Für die korrekte Verwendung der Erweiterung müssen Sie zunächst das mitgelieferte Basis TypoScript 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 TypoScript

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

Eigenschaft	Beschreibung
listPid	Seiten-Uid für die Darstellung der Liste / Suchergebnisse
singlePid	Seiten-Uid für die Darstellung der Detailansicht
formFields	komma-separierte Liste der Felder, die im Formular angezeigt werden sollen
form.kilometers	komma-separierte Liste der Kilometer-Optionen
list.itemsPerPage	Anzahl der Einträge pro Seite
additionalCssFile	Einbindung einer zusätzlichen CSS-Datei im Header. Betrifft allen Seiten auf dem Plugin eingebunden wird.
additionalJsFile	Einbindung einer zusätzlichen JavaScript-Datei im Header. Betrifft allen Seiten auf dem Plugin eingebunden wird.
map.pathToActiveMarker	Pfad zum Icon für den aktiven Marker in der Google Maps Karte
map.pathToDefaultMarker	Pfad zum Icon für die Marker in der Google Maps Karte
map.apiKey	Api-Key für die Einbindung von Google Maps

2.3 Templates

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

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

2.3.1 Templates ändern oder bearbeiten

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

2.3.1.1 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_bzgaberatungsstellensuche {
        view {
                templateRootPaths >
                templateRootPaths {
                        0 = EXT:bzga_beratungsstellensuche/Resources/Private/
↪Templates/
                        1 = EXT:your_sitepackage/Resources/Private/Templates/
                }
                partialRootPaths >
                partialRootPaths {
                        0 = EXT:bzga_beratungsstellensuche/Resources/Private/Partials/
                        1 = EXT:your_sitepackage/Resources/Private/Partials/
                }
                layoutRootPaths >
                layoutRootPaths {
                        0 = EXT:bzga_beratungsstellensuche/Resources/Private/Layouts/
                        1 = EXT:your_sitepackage/Resources/Private/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.

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

2.3.1.1.1 Den Pfad zu den Templates über die constants anpassen

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

```
plugin.tx_bzgaberatungsstellensuche {
        view {
                templateRootPath = EXT:your_sitepackage/Resources/Private/Templates/
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
                partialRootPath = EXT:your_sitepackage/Resources/Private/Partials/
                layoutRootPath = EXT:your_sitepackage/Resources/Private/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}
```

2.3.2.1 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}"/>
```

2.3.2.2 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>
```

2.3.2.3 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>
```

2.3.2.4 DistanceViewHelper

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

Klasse: `Classes/ViewHelpers/DistanceViewHelper.php`

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

2.3.2.5 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>
```

2.3.2.6 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 TypoScript

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

2.4.1.1 Plugin mittels TypoScript einbinden

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

```
lib.beratungsstellensuche = USER
lib.beratungsstellensuche {
        userFunc = TYPO3\CMS\Extbase\Core\Bootstrap->run
        extensionName = BzgaBeratungsstellensuche
        pluginName = Pi1
        vendorName = BZgA

        switchableControllerActions {
                Entry {
                        1 = form
                }
        }

        settings < plugin.tx_bzgaberatungsstellensuche.settings
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        settings {
                formFields = location,kilometers
                listPid = 12345
        }
}

[globalVar = GP:tx_bzgaberatungsstellensuche_pi1|__trustedProperties = /\w+/ ]
        lib.beratungsstellensuche = USER_INT
[global]
```

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

2.4.1.2 Beratungsstelle in der Breadcrumb aufnehmen

Möchten Sie die Beratungsstelle in Ihrer Breadcrumb mit aufnehmen, benutzen Sie den folgenden TypoScrip-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 Beratungsstellen title if on single view
        20 = RECORDS
        20 {
                stdWrap.if.isTrue.data = GP:tx_bzgaberatungsstellensuche_pi1|entry
                dontCheckPid = 1
                tables = tx_bzgaberatungsstellensuche_domain_model_entry
                source.data = GP:tx_bzgaberatungsstellensuche_pi1|entry
                source.intval = 1
                conf.tx_bzgaberatungsstellensuche_domain_model_entry = TEXT
                conf.tx_bzgaberatungsstellensuche_domain_model_entry {
                        field = title
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
                htmlSpecialChars = 1
            }
            stdWrap.wrap = <li>|</li>
            stdWrap.required = 1
        }
}
```

2.4.1.3 Title-Tag für die Detailansicht anpassen

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

2.4.1.3.1 Nur mittels TypoScript

```
[globalVar = GP:tx_bzgaberatungsstellensuche_pi1|entry > 0]

config.noPageTitle = 2

temp.title = RECORDS
temp.title {
        dontCheckPid = 1
        tables = tx_bzgaberatungsstellensuche_domain_model_entry
        source.data = GP:tx_bzgaberatungsstellensuche_pi1|entry
        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.title

[global]
```

2.4.1.3.2 ViewHelper benutzen

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

2.4.2 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.linkhandler.txt`

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**

2.4.3 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.linkvalidator.txt`

2.4.4 Sitemap

Sollte die Extension **seo** 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/?type=1533906435

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

2.4.5 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.

Bemerkung: 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.

KAPITEL 3

Redakteur

3.1 Der Anfang

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.

3.1.2.1 Detailseite

1. Legen Sie einen 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. Sieher 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.

3.1.2.2 Listenansicht

1. Legen Sie einen 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 TypoScript-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 TypoScript-Eigenschaften gesetzt werden.
5. Speichern Sie nach erfolgreicher Konfiguration das Plugin-Inhaltselement ab.

3.1.2.3 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.

Hinweis: 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 TypoScript 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.

3.2.2 Beratungsart

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

KAPITEL 4

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_locallang_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.

4.2.1.1 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::cla
↪s
↪

// 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(
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
                'extendedVariables' => $variables,
                );
        }
}
```

Hinweis: 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.

4.2.2.1 Domain/Repository/EntryRepository.php findDemanded

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

4.2.2.1.1 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

$GLOBALS['TYPO3_CONF_VARS']['EXT']['bzga_beratungsstellensuche']['Domain/Repository/
↪EntryRepository.php']['findDemanded'][$_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 */
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
                $constraints[] = $query->like('keywords', '%unabhängig%');
        }
}
```

Hinweis: 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/**

Hinweis: 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.

Warnung: 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`.

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

Es werden grundsätzlich 2 Dateien benötigt.

4.3.1.1 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,
);
```

4.3.1.2 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
```

4.3.2.1 Eigene Klasse

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

```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;
        }
}
```

4.3.2.2 Clear system cache

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

Hinweis: 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.

4.3.2.3 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.

Hinweis: Zum Thema Slots schauen Sie auch hier: *Signals*.

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

```php
<?php

// Extend name converter
$signalSlotDispatcher->connect(
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
↪\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,
                );
        }
}
```

Hinweis: 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.