

TYPO3 Form Extension

Inhalt:

- Form Framework Eigener Form Definition Ordner
- Form Backend vs. YAML
- Styling
- Eigene Finishers - PDF-Finisher, EmailFinisher
- Eigene Form Elements
- Eigenes Email Template
- Eigene Summary Page / Zwischen-Seite Einfügen und Überladen
- Generelle Vorgehensweise beim adaptieren von Standart Form Core Elements

Form Framework Eigener Form Definition Ordner

```
plugin.tx_form {  
    settings {  
        yamlConfigurations {  
            100 = EXT:firework_template/Configuration/Form/CustomFormSetup.yaml  
        }  
    }  
  
    module.tx_form {  
        settings {  
            yamlConfigurations {  
                100 = EXT:firework_template/Configuration/Form/CustomFormSetup.yaml  
            }  
        }  
    }  
}
```

In der Datei CustomFormSetup.yaml schreiben wir dann:

```
TYP03:  
CMS:  
Form:  
persistenceManager:  
allowedExtensionPaths:  
20: 'EXT:firework_template/Resources/Private/Form/form_definitions/'  
21: 'EXT:firework_template/Classes/Finishers/'  
allowSaveToExtensionPaths: true  
allowDeleteFromExtensionPaths: true
```

Form Backend vs. YAML

Kontaktformular

Anrede *

Bitte wählen

Vorname

Nachname *

Firma

Straße/Nr.

PLZ *

Ort

E-Mail *

Betreff

Nachricht *

Mit dem Absenden des Kontaktformulars erkläre ich mich damit einverstanden, dass meine personenbezogenen Daten gemäß Ihrer Datenschutzerklärung zum Zwecke der Bearbeitung meiner Anfrage verarbeitet werden. *

Kopie an meine E-Mail-Adresse senden

Absenden

The screenshot shows a software interface for managing contact forms. On the left, a tree view lists fields: Anrede (Einfachauswahl), Vorname (Text), Nachname (Text), Firma (Text), Straße/Nr. (Text), PLZ (Text), Ort (Text), E-Mail (E-Mail Adresse), Betreff (Text), Nachricht (Textfeld), Mit dem Absenden des Kontaktformulars (checkbox), and Kopie an meine E-Mail-Adresse senden (checkbox). A button 'Neuen Schritt erstellen' is at the bottom. On the right, a preview window titled 'Schritt 1 von 1' shows the contact form with fields: Anrede * (dropdown with 'Frau' and 'Herr'), Vorname (text input), Nachname * (text input), Firma (text input), Straße/Nr. (text input), PLZ * (text input), and Ort (text input). The preview window has navigation buttons at the top.

renderingOptions:

submitButtonLabel: Absenden

type: Form

identifier: kontaktformular

label: Kontaktformular

prototypeName: standard

renderables:

- **renderingOptions:**

- **previousButtonLabel:** 'Vorherige Seite'

- **nextButtonLabel:** 'Nächster Schritt'

- **type:** Page

- **identifier:** page-1

- **label:** Kontaktformular

- **renderables:**

- **properties:**

- **options:**

- **Frau:** Frau

- **Herr:** Herr

- **prependOptionLabel:** 'Bitte wählen'

- **fluidAdditionalAttributes:**

- **required:** required

- **validationErrorMessages:**

- **code:** 1221560910

- **message:** 'Bitte wählen Sie eine Anrede aus'

- **code:** 1221560718

- **message:** 'Bitte wählen Sie eine Anrede aus'

- **code:** 1347992400

- **message:** 'Bitte wählen Sie eine Anrede aus'

- **code:** 1347992453

```
    message: 'Bitte wählen Sie eine Anrede aus'
type: SingleSelect
identifier: singleselect-1
label: Anrede
validators:
  - identifier: NotEmpty

  defaultValue: ''
type: Text
identifier: vorname
label: Vorname
properties:
  elementClassAttribute: 'kontaktvorname'
validators:
  - identifier: Alphanumeric

  defaultValue: ''
type: Text
identifier: text-2
label: Nachname
properties:
  fluidAdditionalAttributes:
    required: required
  validationErrorMessage:
    - code: 1221560910
      message: 'Bitte geben Sie Ihren Nachnamen ein'

    - code: 1221560718
      message: 'Bitte geben Sie Ihren Nachnamen ein'

    - code: 1347992400
      message: 'Bitte geben Sie Ihren Nachnamen ein'

    - code: 1347992453
      message: 'Bitte geben Sie Ihren Nachnamen ein'
validators:
  - identifier: NotEmpty
  - identifier: Alphanumeric

  defaultValue: ''
type: Text
identifier: text-3
label: Firma
validators:
  - identifier: Alphanumeric

  defaultValue: ''
type: Text
identifier: text-4
label: Straße/Nr.
validators:
```

```
- identifier: Alphanumeric
```

```
- identifier: Integer
```

```
- defaultValue: "
```

```
  type: Text
```

```
  identifier: text-5
```

```
  label: PLZ
```

```
  properties:
```

```
    fluidAdditionalAttributes:
```

```
    required: required
```

Styling

Generell verwendet das Form Framework Bootstrap. Das heißt Styling über Bootstrap Klassen möglich.

In vielen Fällen benötigt man aber eine eigene CSS Klasse für ein Element.

Das geht in dem direkt in der YAML Datei für das entsprechende Form Element folgendes hinzufügt.

```
properties:
```

```
  elementClassAttribute: 'MeineCSSKlasse'
```

Hierbei lassen sich alle Elemente ausser der Submit Button Stylen. Der Grund ist, der Submit Button hat ein eigenes Fluid Partial.

TIPP: Da es häufig vorkommt das Formular insbesondere bei KMP sich häufig unterscheiden, kann um das ganze Formular ein Frame gewrapped werden, so das hierüber das Styling im CSS selektiert werden kann.

Das geht in der TSConfig:

```
TCEFORM.tt_content.frame_class.addItems {  
  formxyz= eigenerformframe  
}
```

Eigene Finisher:

TYPO3:

CMS:

Form:

```
persistenceManager:  
    allowedExtensionPaths:  
        20: 'EXT:firework_template/Resources/Private/Form/form_definitions/'  
        21: 'EXT:firework_template/Classes/Finishers/'  
    allowSaveToExtensionPaths: true  
    allowDeleteFromExtensionPaths: true  
prototypes:  
    standard:  
        finishersDefinition:  
            PdfFinisher:  
                implementationClassName: 'Firework_template\Finishers\PdfFinisher'  
                FormEngine:  
                    label: 'Pdf Finisher'  
            EmailFinisher:  
                implementationClassName: 'Firework_template\Finishers\KmpmailFinisher'  
                FormEngine:  
                    label: 'Email Finisher New'
```

Konkret heißt das, im Ordner Classes liegt eine Datei PDFFinisher.php.

```
<?php
```

```
/*
```

```
Author: Markus Müllenborn-Pitzen
```

```
*/
```

```
// Let register This Namespace for implementationClassName: 'Werbeagentur4you\SiteTemplate\Finishers\PdfFinisher'  
namespace Firework_template\Finishers;
```

```
use TYPO3\CMS\Core\Database\ConnectionPool;  
use TYPO3\CMS\Core\Utility\GeneralUtility;  
use TYPO3\CMS\Form\Domain\Finishers\AbstractFinisher;
```

```
class PdfFinisher extends \TYPO3\CMS\Form\Domain\Finishers\AbstractFinisher  
{
```

```
/**  
 * @var string  
 */  
protected $shortFinisherIdentifier = 'PdfFinisher';
```

function executeInternal()
muss immer vorhanden
sein! Daneben können
weitere angelegt werden.

```
// Execute Internal is the base function that always have to call first.  
protected function executeInternal(): void  
{
```

```
$formfields=$this->finisherContext->getFormValues();
```

```

//var_dump($formfields);
//exit(0);

$fname=$formfields['text-1'];
$fstelle=$formfields['text-2'];

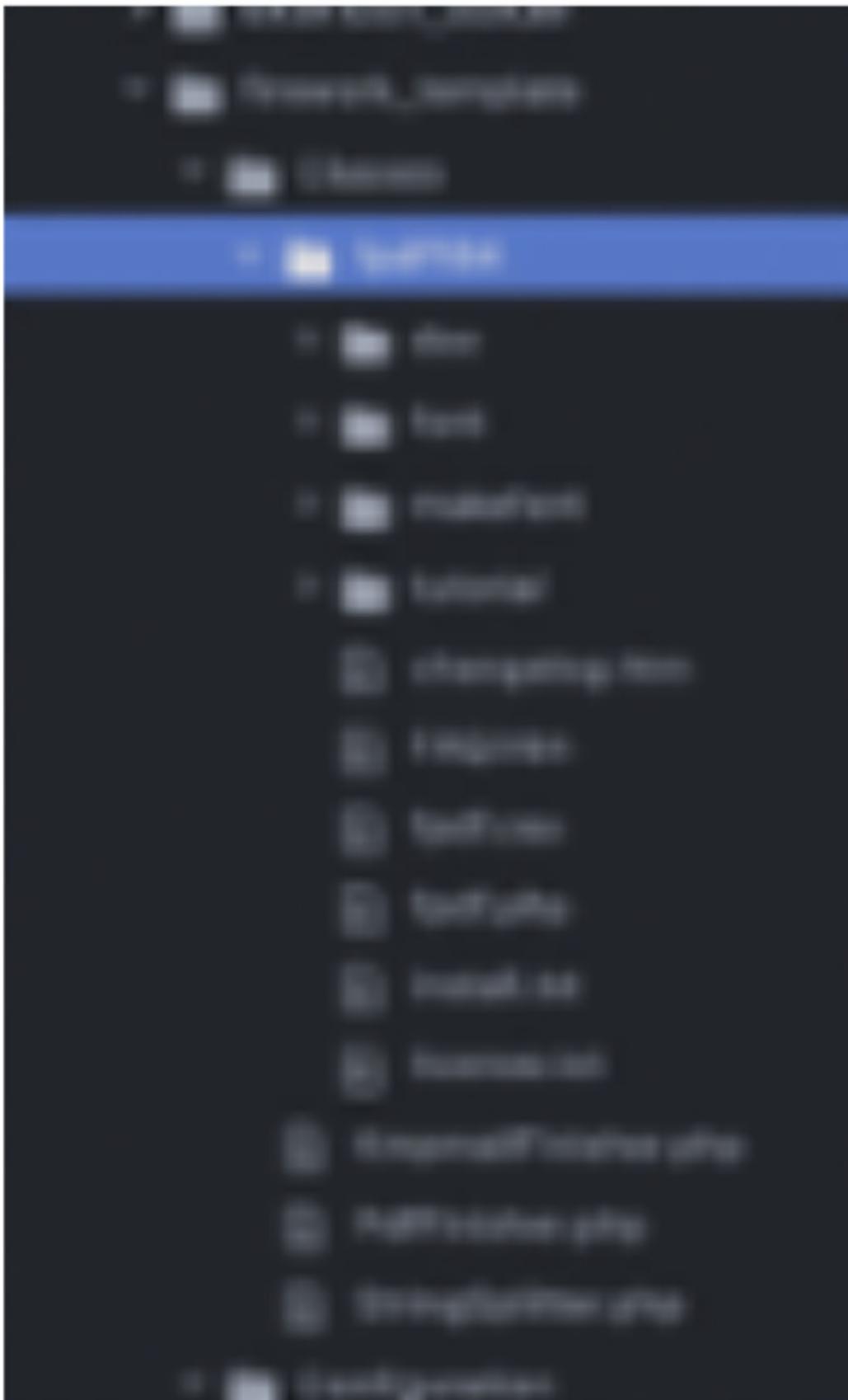
$pdf = new \Fouryou\Fpdf\FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'Eingegebener Name: '.$fname,0,1);
$pdf->Cell(40,10,'Eingegebene Emailadresse: '.$fstelle,0,1);

$pdf->Output(D);

}

}
?>

```



In der Datei fpdf.php

```

<?php

namespace Fouryou\Fpdf;
/*
 * FPDF
 *
 */

```

Da die Autoloader Funktionalität vorliegt reicht es hier den namespace anzupassen. miro

Eigene Form Elements:

In der CustomFormSetup.yaml muss der folgende Eintrag gemacht werden.

```
formElementsDefinition:  
  Form:  
    renderingOptions:  
      partialRootPaths:  
        1505042806: 'EXT:firework_template/Resources/Private/Form/FormElements/'  
      templateRootPaths:  
        1505042806: 'EXT:firework_template/Resources/Private/Form/FormElements/'  
  
  Anschrift:  
    __inheritances:  
      10: 'TYPO3.CMS.Form.prototypes.standard.formElementsDefinition.StaticText'  
  formEditor:  
    label: 'Anschrift'  
    groupSorting: 150
```

In der Datei Anschrift.html kann dann valides Fluid verwendet werden.

Das Einzige was am Anfang stehen muss ist <formvh:renderRenderable

.....

Hier kann dann auch auf TypoScript Libs zugegriffen werden.

```
<html xmlns:f="http://typo3.org/ns/TYPO3/CMS/Fluid/ViewHelpers" xmlns:formvh="http://typo3.org/ns/TYPO3/CMS/Form/ViewHelpers">  
<formvh:renderRenderable renderable="{element}">  
  <f:render partial="Field/Field" arguments="{element: element, doNotShowLabel: 1}">  
    <div class="form-anschrift">  
      <p class="fpersonr">  
        <f:cObject typoscriptObjectPath="lib.personalnr" />  
      </p>  
      <p class="fpfirmennameone">  
        <f:cObject typoscriptObjectPath="lib.firmenname1" />  
      </p>  
      <p class="fpfirmennametwo">  
        <f:cObject typoscriptObjectPath="lib.firmenname2" />  
      </p>  
      <p class="fgender">  
        <f:cObject typoscriptObjectPath="lib.gender" />  
        <f:cObject typoscriptObjectPath="lib.titel" />  
        <f:cObject typoscriptObjectPath="lib.firstname" />  
        <f:cObject typoscriptObjectPath="lib.lastname" />  
      </p>  
      <p class="fstreet">  
        <f:cObject typoscriptObjectPath="lib.strasse" />  
      </p>  
      <p class="fort">  
        <f:cObject typoscriptObjectPath="lib.plz" />  
        <f:cObject typoscriptObjectPath="lib.stadt" />  
      </p>  
    </div>  
  </f:render>  
</formvh:renderRenderable>  
</html>
```

Eigenes Email Template:

In der Formular Definition - yaml Datei muss der folgende Code hinzugefügt werden:

```
options:
  subject: KMP-Bestellformular
  recipients:
    mm30091982@gmail.com: Markus
  senderAddress: erik.dobler@4you-werbeagentur.de
  senderName: ''
  addHtmlPart: true
  attachUploads: true
  templateName: '{@format}.html'
  templateRootPaths:
    20: 'EXT:firework_template/Resources/Private/Form/Emails/Visitenkartenbestell/'
  layoutRootPaths:
    20: 'EXT:firework_template/Resources/Private/Form/Emails/Visitenkartenbestell/'
  translation:
    language: Standard
    useFluidEmail: true
    title: ''
  identifier: EmailToReceiver
```

Wichtig sind hierbei die Teile addHTMLPart, templateName und natürlich der Pfad zu den Templates.

In dem EmailTemplate ansich können wir dann auch wieder Fluid verwenden.

```
{namespace site=Firework_template\ViewHelper}
{namespace v=FluidTYPO3\Vhs\ViewHelpers}
<f:layout name="SystemEmailNew" />
<f:section name="Title">
  <h3>{title}</h3>
  <f:debug>{_all}</f:debug>
</f:section>
<f:section name="Main">
  <p>Der/Die freie Mitarbeiter/in
    {form.elementsByIdentifier.formValues.visitenkarte_anzahl}
    <f:cObject typoscriptObjectPath="lib.lastname" />
    hat eine Bestellung abgeschickt:
  </p>
  <f:for each="{v:iterator.explode(content: '{form.formState.formValues.hiddentoken}', glue: ';')}" as="exploded"> {exploded}<br> </f:for>

  <br>
  <span class="bestellformular-text"><f:cObject typoscriptObjectPath="lib.personalnr" /></span>
  <u>Bestellung:</u>
  <br>
```

```
<span class="bestellformular-  
text"><b>{form.formState.formValues.visitenkarte_anzahl}</b> Paket/e Visitenkarten à EUR  
29,90 netto zzgl. MwSt. frei Haus</span>
```

<u>Anmerkung:</u>

```
<f:for each="{v:iterator.explode(content: '{form.formState.formValues.hiddentoken}',  
glue: ';')}" as="exploded"> {exploded}<br> </f:for>
```



```
<span class="bestellformular-  
text">{form.formState.formValues.visitenkarte_anmerkung}</span>  
</f:section>
```

Achtung
exploded kein
Standartviewhel-
per->VHS+Eigen
er ViewHelper

Eigene Summary Page/ Standart Sysextr Teile überladen

In der CustomFormSetup.yaml muss am Ende der Grundlegende Pfad angegeben werden.

```
bestellformular:  
  __inheritances:  
    10: 'TYPO3.CMS.Form.prototypes.standard'  
  formElementsDefinition:  
    Form:  
      renderingOptions:  
        templateRootPaths:  
          100: 'EXT:firework_template/Resources/Private/Form/Emails/Ownstep'  
        partialRootPaths:  
          100: 'EXT:firework_template/Resources/Private/Form/Emails/Ownstep'  
        layoutRootPaths:  
          100: 'EXT:firework_template/Resources/Private/Form/Emails/Ownstep'
```

In dem Ordner Ownstep liegen dann unsere Ganzen Individuellen SummaryPages.

Möchte wir nun unserer eigene SummaryPage verwenden muss wir dies in der Form Definition angeben.

```
renderingOptions:  
  previousButtonLabel: 'Vorherige Seite'  
  nextButtonLabel: 'Nächster Schritt'  
type: SummaryPage  
identifier: summarypage-1  
label: Zusammenfassung
```

Die Summarypage.html sähe dann bspw. so aus:

```
<html xmlns:f="http://typo3.org/ns/TYPO3/CMS/Fluid/ViewHelpers"  
xmlns:formvh="http://typo3.org/ns/TYPO3/CMS/Form/ViewHelpers" data-namespace-
```

```

typo3-fluid="true">
<formvh:renderRenderable renderable="{page}">
  <fieldset class="form-group">
    <f:if condition="{page.label}">
      <legend>{formvh:translateElementProperty(element: page, property: 'label')}</legend>
    </f:if>
    <div class="table-responsive">

<h2>Eigens Forms</h2>
  <table class="table">
    <formvh:renderAllFormValues renderable="{page.rootForm}" as="formValue">
      <tr>
        <f:if condition="{formValue.isSection}">
          <f:then>
            <td colspan="2"><b>{formvh:translateElementProperty(element: formValue.element, property: 'label')}</b></td>
          </f:then>
          <f:else>
            <td class="summary-table-first-col">{formvh:translateElementProperty(element: formValue.element, property: 'label')}</td>
            <td>
              <f:if condition="{formValue.value}">
                <f:then>
                  <f:if condition="{0: formValue.element.type} == {0: 'ImageUpload'}">
                    <f:then>
                      <f:image image="{formValue.value}" maxWidth="{formValue.element.properties.imageMaxWidth}" maxHeight="{formValue.element.properties.imageMaxHeight}" alt="{formvh:translateElementProperty(element: formValue.element, property: 'altText')}"/>
                    </f:then>
                    <f:else>
                      <f:if condition="{formValue.isMultiValue}">
                        <f:then>
                          <ul>
                            <f:for each="{formValue.processedValue}" as="value">
                              <li>{value}</li>
                            </f:for>
                          </ul>
                        </f:then>
                        <f:else>

```

<f:format.nl2br>{formValue.processedValue}</f:format.nl2br>

```

        </f:else>
        </f:if>
        </f:else>
        </f:if>
        </f:then>
        <f:else>
        -
        </f:else>
        </f:if>
        </td>
        </f:else>
        </f:if>
    </tr>
</formvh:renderAllFormValues>
</table>
</div>
</fieldset>
</formvh:renderRenderable>
</html>

```

Um hier eine valide Datei zu bekommen empfiehlt es sich immer mit dem Partial aus der Sysex Form zu beginnen. Das heißt den Inhalt zu kopieren. Die Partials die kopiert werden können liegen unter:

/html/typo3/typo3_src-11.5.17/typo3/sysex/form/Resources/Private/Frontend/Partials/

Modifikation von Standartformelementen:

Im Grunde gleiche vorgehensweise wie oben. Als Beispiel schauen wir uns die Checkbox an.

Häufig hat man eine Checkbox wo im Labeltext ein Link rein soll zum Datenschutz.

Wie macht man das?

Ganz einfach!

in dem Form muss der Partial Part angegeben werden:

```

partialRootPaths:
  1505042806: 'EXT:site_template/Resources/Private/Extensions/Form/Partials/
  ...'

```

Nun können wir direkt in der YAML Datei die Checkbox hinzufügen.

```

  properties:

```

```

properties:
  pageUid: '35'
  linkText: Datenschutzerklärung
  linkText2: 'zum Zwecke der Bearbeitung meiner Anfrage verarbeitet werden.'
  fluidAdditionalAttributes:
    required: required
  type: LinkedCheckbox
  identifier: linkedcheckbox-1
  label: 'Mit dem Absenden des Kontaktformulars erkläre ich mich damit einverstanden,
  validators:
  -
    identifier: NotEmpty

```

In diesem Fall verwenden wir also das Partial LinkedCheckbox

```

<html xmlns:f="http://typo3.org/ns/TYPO3/CMS/Fluid/ViewHelpers"
      xmlns:formvh="http://typo3.org/ns/TYPO3/CMS/Form/ViewHelpers" data-namespac
fluid="true">
  <formvh:renderRenderable renderable="{element}">
    <f:render partial="Field/Field" arguments="{element: element, doNotShowLabel: 1
contentAs="elementContent">
      <div class="form-check">
        <label class="{element.properties.elementClassAttribute} form-check-label">
          <f:form.checkbox
            property="{element.identifier}"
            id="{element.uniqueIdentifier}"
            class="{element.properties.elementClassAttribute}"
            value="{element.properties.value}"
            errorClass="{element.properties.elementErrorClassAttribute}"
            additionalAttributes="{formvh:translateElementProperty(element: elem
property: 'fluidAdditionalAttributes')}">
            />
            <span>{formvh:translateElementProperty(element: element, property: 'label
<f:link.typolink parameter="{element.properties.pageUid}"
target="_blank">{formvh:translateElementProperty(element: element, property:
'linkText')}</f:link.typolink> {formvh:translateElementProperty(element: element, prop
'linkText2')}</f:if> <f:render partial="Field/Required"
/></f:if></span>
          </label>
        </div>
      </f:render>
    </formvh:renderRenderable>
  </html>

```

Bonus Email an Absender:

Direkt im Form Editor im Typo3 Backend können wir zwar direkt einen zweiten Finisher Email hinzufügen, aber wie bekommen wir es hin das er nur gefeuert wird wenn eine Checkbox ausgewählt ist?

Die Antwort ist mit einer Condition in der Yaml Datei.

```
- identifier: variant-kopie
  condition: 'formValues["checkbox-2"] == 1'
  finishers:
    - options:
        subject: 'Kopie Ihrer Nachricht an Lohnsteuerberatungsverbund e.V.-Lohnsteuerhilfverein-'
        recipientAddress: '{email-1}'
        recipientName: '{text-1} {text-2}'
        senderAddress: info@steuerverbund.de
        senderName: 'Lohnsteuerberatungsverbund e.V. -Lohnsteuerhilfverein-'
        replyToAddress: ''
        carbonCopyAddress: ''
        blindCarbonCopyAddress: ''
        format: html
        attachUploads: false
        templateName: '{@format}.html'
        templateRootPaths:
          1505042806: 'EXT:site_template/Resources/Private/Extensions/Form/Templates/Finishers/Email/Sender/'
        partialRootPaths:
          1505042806: 'EXT:site_template/Resources/Private/Extensions/Form/Partials/'
  identifier: EmailToSend
```

Bonus: Backend Validation

Das Form Framework bringt schon eigene Validatoren mit, wie etwa Alphanumerisch oder Numerisch. Leider sind die vorhandenen Validatoren für unsere Zwecke nicht ausreichend.

In den meisten Fällen helfen wir uns hier mit Frontend Validation JQUERY.

Problem: Schalte im Browser JS ab und du kommst durch.

In Form Setup muss folgendes hinzugefügt werden.

```
prototypes:
standard:
  validatorsDefinition:
    Custom:
      implementationClassName: '\firework_template\Domain\Validation\CustomValidator'
```

Beispiel: Prüfen ob gültige Anrede in der Form Herr oder Frau.

In der zugehörigen PHP Datei könnte man dann folgendes machen:

```
<?php
```

```
namespace firework_template\Domain\Validation\CustomValidator;  
  
use TYPO3\CMS\Extbase\Validation\Validator\AbstractValidator;  
  
class TitleValidator extends AbstractValidator  
{  
    protected function isValid($value)  
    {  
        // $value is the title string  
        if (count(explode(':', $value)) >= 4) {  
            return;  
        }  
        $this->addError('Bitte man oder Frau eingeben', 1221559976);  
    }  
}
```

DAS WAR's. Ab hier bitte Applaus ;)