

ENTWICKLUNG EINER FORMULARANWENDUNG MIT KOMPATIBILITÄTSVALIDIERUNG DER EINFACH- UND MEHRFACHAUSWAHL-EINGABEFELDER

Vorgelegt von:

Alexander Johr

Meine Adresse

Erstprüfer: Prof. Jürgen Singer Ph.D.
Zweitprüfer: Prof. Daniel Ackermann
Datum: 02.11.2020

Teil I

Implementierung

0.1 Schritt 3

In diesem Schritt soll die grundlegende Validierungsfunktion hinzugefügt werden. Maßnahmen, die als abgeschlossen markiert sind, dürfen keine leeren Eingabefelder enthalten und der Maßnahmentitel darf nicht doppelt belegt sein. `Flutter` stellt das Widget `Form` für die Validierung von Eingabefeldern bereit.

0.2 Einfügen des Form-Widgets

Das Widget `Form` ist ein Container, welcher die Validierung für alle Kinderelemente des Typs `FormField` ausführt. Damit es alle Eingabefelder im Formular umgibt, wird es zwischen dem `WillPopScope` und dem `Stack` eingefügt (Listing. 1, Z. 168). Darüber hinaus wird die Funktion `saveRecord` durch `saveAndGoBackToOverviewScreenIfValid` ersetzt (Z. 167), welche ebenfalls in diesem Schritt implementiert wird. Das `Form`-Widget muss über einen `key` registriert werden (Z. 169), damit auf die Validierungsfunktionen zurückgegriffen werden kann.

```
158 return Scaffold(  
159   appBar: AppBar(  
160     title: const Text('Maßnahmen Detail'),  
161   ),  
162   body: WillPopScope(  
163     onWillPop: saveAndGoBackToOverviewScreenIfValid,  
164     child: Form(  
165       key: formKey,  
166       child: Stack(  

```

Listing 1: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

Die Erstellung des `formKey` findet zu Beginn der `build`-Methode des Eingabeformulars statt (Listing. 2, Z. 20). Der `GlobalKey` identifiziert ein Element, welches durch ein Widget gebaut wurde, über die gesamte Applikation hinweg. Es erlaubt darüber hinaus auf das `State`-Objekt zuzugreifen, welches mit dem `StatefulWidget` verknüpft ist. Ohne Angabe eines Typparameters kann nur Zugriff auf Funktionen des Typs `State` gewährt werden. Doch die gewünschte Methode `validate` ist nur Teil des Typs `FormState`. Damit das Element, welches über den `GlobalKey` registriert wurde, auch den `FormState` liefert, kann der entsprechende Typparameter `<FormState>` bei der Erstellung des `GlobalKey` übergeben werden.

```
17 Widget build(BuildContext context) {  
18   final vm = AppState.of(context).viewModel;  
19   final model = AppState.of(context).model;  
20   final formKey = GlobalKey<FormState>();  

```

Listing 2: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

0.3 Validierung des Maßnahmentitels

Das Eingabefeld für den Maßnahmen-Titel ist ein `TextFormField` (Listing. 3, Z. 99). Es erbt vom Typ `FormField` und wird daher mit dem Väterelement `Form` verknüpft. Es beinhaltet bereits einen Parameter für die Validierungsfunktion namens `validator` (Z. 104). Die übergebene Funktion erhält im ersten Parameter den für das Textfeld eingetragenen Wert. Die Funktion soll `null` zurückgeben, wenn keine Fehler in der Validierung geschehen sind. In

jedem anderen Fall soll der Text zurückgegeben werden, der als Fehlermeldung angezeigt werden soll.

```
94 Widget createMassnahmenTitelTextFormField() {
95   final focusNode = FocusNode();
96   return Card(
97     child: Padding(
98       padding: const EdgeInsets.all(16.0),
99       child: TextFormField(
100         focusNode: focusNode,
101         initialValue: vm.massnahmenTitel.value,
102         decoration: const InputDecoration(
103           hintText: 'Maßnahmentitel', labelText: 'Maßnahmentitel'),
104         validator: (title) {
105           if (title == null || title.isEmpty) {
106             focusNode.requestFocus();
107             return 'Bitte Text eingeben';
108           }
109           var massnahmeTitleDoesAlreadyExists =
110             model.storage.value.massnahmen.any((m) =>
111               m.guid != vm.guid.value &&
112               m.identifikatoren.massnahmenTitel ==
113                 vm.massnahmenTitel.value);
114
115           if (massnahmeTitleDoesAlreadyExists) {
116             focusNode.requestFocus();
117             return 'Dieser Maßnahmentitel ist bereits vergeben';
118           }
119           return null;
120         },
121         onChanged: (value) {
122           vm.massnahmenTitel.value = value;
123         },
124       ),
125     ),
126   );
127 }
```

Listing 3: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

Sollte der Parameter `null` sein oder aber ein leerer String (Z. 105), so wird die entsprechende Fehlermeldung `'Bitte Text eingeben'` angezeigt (Z. 107). Damit der Benutzer direkt zu dem fehlerhaften Eingabefeld geführt wird, kann ein Objekt der Klasse `FocusNode` verwendet werden. Er wird vor der Konstruktion der Karte erstellt (Z. 95) und dem Parameter `focusNode` des `TextFormField` übergeben (Z. 100). Sollte ein Fehler bei der Validierung gefunden werden, kann mit der Methode `requestFocus` angeordnet werden, den Cursor in das betreffende Feld zu setzen (Z. 106). Das sorgt auch dafür, dass das Eingabefeld in den sichtbaren Bereich gerückt wird.

Sollte das Textfeld nicht leer sein, so soll noch überprüft werden, ob der Maßnahmen-Titel bereits vergeben ist. Über das Model kann die Liste der Maßnahmen angefordert werden (Z. 110). Die Funktion `any` akzeptiert als Argument eine Funktion, die für alle Elemente der Liste ausgeführt wird (Z. 110-113). Wenn die Rückgabe der Funktion auch nur in einem Fall `true` ist, so evaluiert auch `any` mit `true`. Andernfalls ist die Rückgabe `false`. Die anonyme Funktion schließt zunächst den Vergleich mit derselben Maßnahme aus, welche sich gerade in Bearbeitung befindet. Der Vergleich der `guid` ist dafür ausreichend. Sollte es eine andere Maßnahme geben, welche den gleichen Titel hat (Z. 112-113), so wird Die lokale Variable `massnahmeTitleDoesAlreadyExists` auf `true` gesetzt. Der Benutzer bekommt die entsprechende Fehlermeldung `'Dieser Maßnahmentitel ist bereits vergeben'` zu lesen 117. Wenn keine der beiden Fallunterscheidungen das `return`-Statement (Z. 107, 117) auslöst,

so erfolgt schließlich die Rückgabe von `null`. In dem Kontext der `validator`-Funktion bedeutet die Rückgabe von `null`, dass die Validierung erfolgreich war.

Das `Form`-Widget validiert lediglich Kindelemente vom Typ `FormField`. Dementsprechend wird das Widget `SelectionCard` nicht in die Validierung miteinbezogen. Es erbt nicht von `FormField`. Es wäre möglich, eine weitere Klasse zu erstellen, die von `FormField` erbt und alle Parameter für die Erstellung einer Selektions-Karte wiederverwendet. Doch das würde bedeuten, dass für alle folgenden Schritte jeder weitere Parameter in beiden Konstruktoren der Klassen gepflegt werden müsste. Um der Arbeit leichter folgen zu können, wurde sich für einen anderen, simpleren Weg entschieden: Die Selektionskarte kann ebenso von einem `FormField` umgeben werden (Listing. 4, Z. 132-159), welches die Selektionskarte in der `builder`-Funktion erstellt und an den Parametern nichts ändert, außer einen weiteren hinzuzufügen: der Text für die Fehlermeldung (Z. 158). Der erste Parameter der `builder`-Funktion ist das `State`-Objekt des `FormField`. Es enthält die Getter-Methode `errorText`, die bei gegebenenfalls fehlgeschlagener Validierung die zurückgegebene Fehlermeldung enthält.

```
129 Widget buildSelectionCard<ChoiceType extends Choice>(
130     {required Choices<ChoiceType> allChoices,
131     required BehaviorSubject<ChoiceType?> selectionViewModel}) {
132     return FormField(
133         validator: (_) {
134             Iterable<Choice> choices = {
135                 if (selectionViewModel.value != null) selectionViewModel.value!
136             };
137
138             if (choices.isEmpty) {
139                 return "Feld ${allChoices.name} enthält keinen Wert!";
140             }
141
142             return null;
143         },
144         builder: (field) => SelectionCard<ChoiceType>(
145             title: allChoices.name,
146             allChoices: allChoices,
147             initialValue: {
148                 if (selectionViewModel.value != null)
149                     selectionViewModel.value!
150             },
151             onSelect: (selectedChoice) =>
152                 selectionViewModel.value = selectedChoice,
153             onDeselect: (selectedChoice) => selectionViewModel.value = null,
154             errorText: field.errorText,
155         ));
156 }
```

Listing 4: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

Die anonyme Funktion, die als Argument dem Parameter `validator` übergeben wird (Z. 133-143), erstellt eine temporäre Menge, die den Wert des `selectionViewModel` enthält, wenn dieser nicht `null` ist, andernfalls ist sie eine leere Menge (Z. 134-136). Die `validator`-Funktion gibt eine Fehlermeldung zurück, sollte die Menge leer sein (Z. 138-140). Ist die Menge dagegen gefüllt, so gibt sie `null` zurück, um mitzuteilen, dass die Validierung erfolgreich war (Z. 142).

Der `errorText` wird im Konstruktor der Klasse `SelectionCard` übergeben (Listing. 5, Z. 29). Da er `null` sein darf, ist er mit dem Suffix `?` als Typ mit Null-Zulässigkeit gekennzeichnet (Z. 21).

Durch Einfügen einer `Column` zwischen der `Card` (Listing. 6, Z. 53) und dem `ListTile` (Z.

```

19 final OnSelect<ChoiceType> onSelect;
20 final OnDeselect<ChoiceType> onDeselect;
21 final String? errorText;
22
23 SelectionCard(
24   {required this.title,
25     required Iterable<ChoiceType> initialValue,
26     required this.allChoices,
27     required this.onSelect,
28     required this.onDeselect,
29     this.errorText,
30     Key? key})

```

Listing 5: errorText wird der SelectionCard hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart](#)

57) kann die visuelle Repräsentation der Selektionskarte in der Höhe erweitert werden. Sollte der errorText gesetzt sein (Z. 65), so erscheint unter dem Titel und dem Untertitel eine entsprechende Fehlermeldung (Z. 66-71).

```

53 return Card(
54   child: Column(
55     crossAxisAlignment: CrossAxisAlignment.start,
56     children: [
57       ListTile(
58         focusNode: focusNode,
59         title: Text(title),
60         subtitle: Text(
61           selectedChoices.map((c) => c.description).join(", "),
62         trailing: const Icon(Icons.edit),
63         onTap: navigateToSelectionScreen,
64       ),
65       if (errorText != null)
66         Padding(
67           padding: const EdgeInsets.all(8.0),
68           child: Text(errorText!,
69             style:
70               const TextStyle(fontSize: 12.0, color: Colors.red)),
71         )
72     ],
73   ),
74 );

```

Listing 6: errorText wird ausgegeben, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart](#)

Oberhalb des vorhandenen `FloatingActionButton` wird nun ein weiterer eingefügt, der zum Speichern des Entwurfs mit der Funktion `saveDraftAndGoBackToOverviewScreen` genutzt werden soll (Listing. 7, Z. 210-216). Der ursprüngliche `FloatingActionButton` versucht die Validierung und die anschließende Speicherung der Maßnahme mithilfe der neuen Funktion `saveAndGoBackToOverviewScreenIfValid` (Z. 224).

```

209 children: [
210   FloatingActionButton(
211     mini: true,
212     heroTag: 'save_draft_floating_action_button',
213     child: const Icon(Icons.paste, color: Colors.white),
214     backgroundColor: Colors.orange,
215     onPressed: saveDraftAndGoBackToOverviewScreen,
216   ),
217   const SizedBox(
218     height: 10,
219   ),
220   FloatingActionButton(
221     tooltip: saveMassnahmeTooltip,
222     heroTag: 'save_floating_action_button',
223     child: const Icon(Icons.check, color: Colors.white),
224     onPressed: saveAndGoBackToOverviewScreenIfValid,
225   )
226 ],

```

Listing 7: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

22 void saveDraftAndGoBackToOverviewScreen() {
23   ScaffoldMessenger.of(context)
24     ..hideCurrentSnackBar()
25     ..showSnackBar(
26       const SnackBar(content: Text('Entwurf wird gespeichert ...')));
27
28   var draft = vm.model.rebuild((b) =>
29     b.letzteBearbeitung.letzterStatus = LetzterStatus.bearb.abbreviation);
30
31   model.putMassnahmeIfAbsent(draft);
32   Navigator.of(context).pop();
33 }

```

Listing 8: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

84 Future<bool> saveAndGoBackToOverviewScreenIfValid() {
85   if (inputsAreValidOrNotMarkedFinal()) {
86     saveRecordAndGoBackToOverviewScreen();
87     return Future.value(true);
88   } else {
89     showValidationError();
90     return Future.value(false);
91   }
92 }

```

Listing 9: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

72 bool inputsAreValidOrNotMarkedFinal() {
73   if (vm.letzterStatus.value != LetzterStatus.fertig) {
74     return true;
75   }
76
77   if (formKey.currentState!.validate()) {
78     return true;
79   }

```

Listing 10: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

45 void showValidationError() {
46   ScaffoldMessenger.of(context).showSnackBar(SnackBar(
47     content: Row(
48       children: [
49         Text(
50           'Fehler im Formular trotz Status "${LetzterStatus.fertig.description}"',
51           const SizedBox(width: 4),
52         ElevatedButton(
53           onPressed: saveDraftAndGoBackToOverviewScreen,
54           child: Padding(
55             padding: const EdgeInsets.fromLTRB(4, 4, 8, 4),
56             child: Row(
57               children: const [
58                 Icon(Icons.paste, color: Colors.white),
59                 SizedBox(width: 4),
60                 Text(
61                   "Entwurf speichern?",
62                   style: TextStyle(fontSize: 18.0, color: Colors.white),
63                 ),
64               ],
65             ),
66           ),
67         ),
68       ],
69     )),
70 }

```

Listing 11: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

Teil II

Anhang

A Schritt 2 Anhang

```

5 class FoerderklasseChoice extends Choice {
6   static final oelb = FoerderklasseChoice("oelb", "Ökolandbau");
7   static final azl = FoerderklasseChoice("azl", "Ausgleichszulage");
8   static final ea = FoerderklasseChoice("ea", "Erschwerenausgleich");
9   static final aukm_nur_vns = FoerderklasseChoice("aukm_nur_vns",
10     "Agrarumwelt-(und Klima)Maßnahme: nur Vertragsnaturschutz");
11   static final aukm_ohne_vns = FoerderklasseChoice("aukm_ohne_vns",
12     "Agrarumwelt-(und Klima)Maßnahmen, tw. auch mit Tierwohlaspekten, aber OHNE
13     ↳ Vertragsnaturschutz");
14   static final twm_ziel = FoerderklasseChoice(
15     "twm_ziel", "Tierschutz/Tierwohlmaßnahmen mit diesem als Hauptziel");
16   static final contact =
17     FoerderklasseChoice("contact", "bitte um Unterstützung");
18   FoerderklasseChoice(String abbreviation, String description,
19     {bool Function(Set<Choice> choices)? condition})
20     : super(abbreviation, description);
21 }
22
23 final foerderklasseChoices = Choices<FoerderklasseChoice>({
24   FoerderklasseChoice.oelb,
25   FoerderklasseChoice.azl,
26   FoerderklasseChoice.ea,
27   FoerderklasseChoice.aukm_nur_vns,
28   FoerderklasseChoice.aukm_ohne_vns,
29   FoerderklasseChoice.twm_ziel,
30   FoerderklasseChoice.contact
31 }, name: "Förderklasse");
32
33 class KategorieChoice extends Choice {
34   static final zf_us =
35     KategorieChoice("zf_us", "Anbau Zwischenfrucht/Untersaat");
36   static final anlage_pflege =
37     KategorieChoice("anlage_pflege", "Anlage/Pflege Struktur");
38   static final dungmang = KategorieChoice("dungmang", "Düngemanagement");
39   static final extens = KategorieChoice("extens", "Extensivierung");
40   static final flst = KategorieChoice("flst", "Flächenstilllegung/Brache");
41   static final umwandlg = KategorieChoice("umwandlg", "Nutzungsumwandlung");
42   static final bes_kult_rass = KategorieChoice(
43     "bes_kult_rass", "Förderung bestimmter Rassen / Sorten / Kulturen");
44   static final contact = KategorieChoice("contact", "bitte um Unterstützung");
45
46   KategorieChoice(String abbreviation, String description)
47     : super(abbreviation, description);
48 }
49
50 final kategorieChoices = Choices<KategorieChoice>({
51   KategorieChoice.zf_us,
52   KategorieChoice.anlage_pflege,
53   KategorieChoice.dungmang,
54   KategorieChoice.extens,
55   KategorieChoice.flst,
56   KategorieChoice.umwandlg,
57   KategorieChoice.bes_kult_rass,
58   KategorieChoice.contact
59 }, name: "Kategorie");

```

Listing 12: Die Mengen foerderklasseChoices und kategorieChoices , Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

```

61 class ZielflaecheChoice extends Choice {
62   static final ka = ZielflaecheChoice("ka", "keine Angabe/Vorgabe");
63   static final al = ZielflaecheChoice("al", "AL");
64   static final gl = ZielflaecheChoice("gl", "GL");
65   static final lf = ZielflaecheChoice("lf", "LF");
66   static final dk_sk = ZielflaecheChoice("dk_sk", "DK/SK");
67   static final hff = ZielflaecheChoice("hff", "HFF");
68   static final biotop_le =
69     ZielflaecheChoice("biotop_le", "Landschaftselement/Biotop o.Ä.");
70   static final wald = ZielflaecheChoice("wald", "Wald/Forst");
71   static final contact = ZielflaecheChoice("contact", "bitte um Unterstützung");
72
73   ZielflaecheChoice(String abbreviation, String description)
74     : super(abbreviation, description);
75 }
76
77 final zielflaecheChoices = Choices<ZielflaecheChoice>({
78   ZielflaecheChoice.ka,
79   ZielflaecheChoice.al,
80   ZielflaecheChoice.gl,
81   ZielflaecheChoice.lf,
82   ZielflaecheChoice.dk_sk,
83   ZielflaecheChoice.hff,
84   ZielflaecheChoice.biotop_le,
85   ZielflaecheChoice.wald,
86   ZielflaecheChoice.contact
87 }, name: "Zielfläche");
88
89 class ZieleinheitChoice extends Choice {
90   static final ka = ZieleinheitChoice("ka", "keine Angabe/Vorgabe");
91   static final m3 = ZieleinheitChoice("m3", "m3 (z.B. Gülle)");
92   static final pieces =
93     ZieleinheitChoice("pieces", "Kopf/Stück (z.B. Tiere oder Bäume)");
94   static final gve = ZieleinheitChoice("gve", "GV/GVE");
95   static final rgve = ZieleinheitChoice("rgve", "RGV");
96   static final ha = ZieleinheitChoice("ha", "ha");
97   static final contact = ZieleinheitChoice("contact", "bitte um Unterstützung");
98
99   ZieleinheitChoice(String abbreviation, String description)
100     : super(abbreviation, description);
101 }
102
103 final zieleinheitChoices = Choices<ZieleinheitChoice>({
104   ZieleinheitChoice.ka,
105   ZieleinheitChoice.m3,
106   ZieleinheitChoice.pieces,
107   ZieleinheitChoice.gve,
108   ZieleinheitChoice.rgve,
109   ZieleinheitChoice.ha,
110   ZieleinheitChoice.contact
111 }, name: "Zieleinheit");

```

Listing 13: Die Mengen zielflaecheChoices und zieleinheitChoices, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

```

113 class ZielsetzungLandChoice extends Choice {
114   static final ka = ZielsetzungLandChoice("ka", "keine Angabe/Vorgabe");
115   static final bsch = ZielsetzungLandChoice("bsch", "Bodenschutz");
116   static final wsch = ZielsetzungLandChoice("wsch", "Gewässerschutz");
117   static final asch = ZielsetzungLandChoice("asch", "Spezieller Artenschutz");
118   static final biodiv = ZielsetzungLandChoice("biodiv", "Biodiversität");
119   static final struktkviel =
120     ZielsetzungLandChoice("struktkviel", "Erhöhung der Strukturvielfalt");
121   static final genet_res = ZielsetzungLandChoice("genet_res",
122     "Erhaltung genetischer Ressourcen (Pflanzen, z. B. im Grünland, und Tiere, z. B.
123     ↳ bedrohte Rassen)");
124   static final tsch = ZielsetzungLandChoice(
125     "tsch", "Tierschutz/Maßnahmen zum Tierwohl im Betrieb");
126   static final klima = ZielsetzungLandChoice("klima", "Klima");
127   static final contact =
128     ZielsetzungLandChoice("contact", "bitte um Unterstützung");
129   ZielsetzungLandChoice(String abbreviation, String description)
130     : super(abbreviation, description);
131 }
132
133 final _zielsetzungLandChoices = {
134   ZielsetzungLandChoice.ka,
135   ZielsetzungLandChoice.bsch,
136   ZielsetzungLandChoice.wsch,
137   ZielsetzungLandChoice.asch,
138   ZielsetzungLandChoice.biodiv,
139   ZielsetzungLandChoice.struktkviel,
140   ZielsetzungLandChoice.genet_res,
141   ZielsetzungLandChoice.tsch,
142   ZielsetzungLandChoice.klima,
143   ZielsetzungLandChoice.contact
144 };
145
146 final hauptzielsetzungLandChoices = Choices<ZielsetzungLandChoice>(
147   _zielsetzungLandChoices,
148   name: "Hauptzielsetzung Land");

```

Listing 14: Die Menge hauptzielsetzungLandChoices, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)