

【Power Appsでハッカソンしたいけえ #10】

Power Appsの

バリデーションチェック(エラーチェック)

～PPの城窓から～



PPログ 2024/11/24



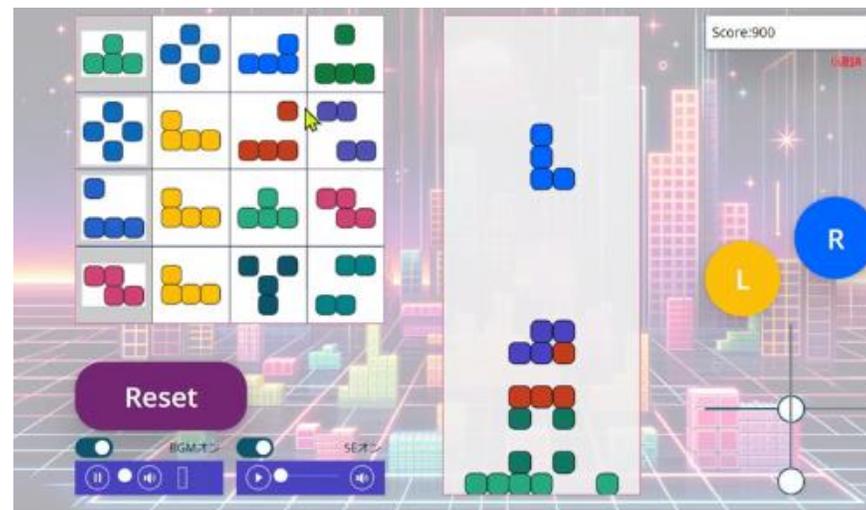
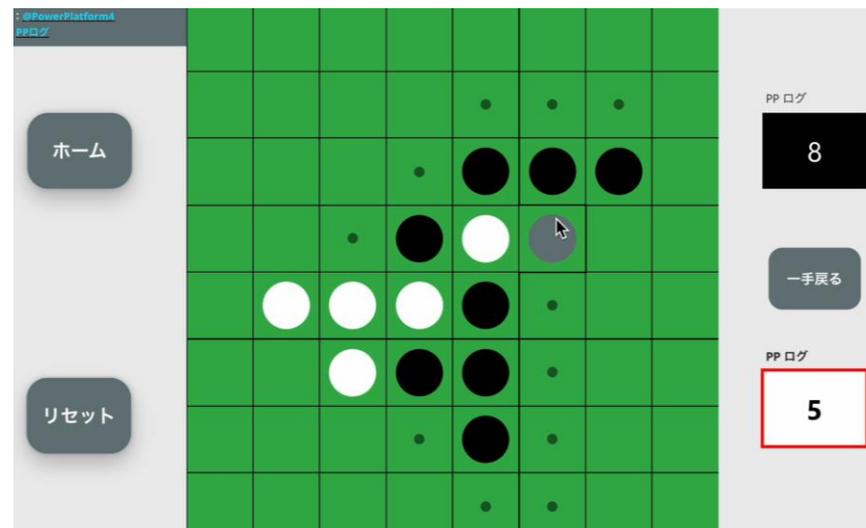
PPログ

X(Twitter) :

<https://twitter.com/PowerPlatform4>

Blog :

<https://powerplatformnikki.com/>





Power Platformの城

イベント

メンバー

資料

B! 0

いいね! 0

グループの説明

「Power Platformの城」は、市民(市民開発者・初心者)を大事にするコミュニティです。市民から王族(プロ)まで、どなたでも参加&登壇募集しております。

ハッシュタグ #PP城

[ツイートする](#) [ツイート検索](#)

Content

- まえおき
- バリデーションチェックの実装パターンを比較
- Errorプロパティを利用したリアルタイムチェック
- コントロールからフォーカスアウト時にチェック
- SubmitForm前にIfでチェック

まえおき

バリデーションチェックの実装には工数がかかるのでできるだけデフォルトのままにしておくことを推奨

列

列には、リスト内の各アイテムについての情報が保存されます。現在、このリストでは次の列を使用できます。

列 (クリックして編集)	種類
タイトル	1 行テキスト
価格	数値
画面サイズ	1 行テキスト
発売日	日付と時刻
スペック	1 行テキスト
容量	選択肢
販売中	はい/いいえ
user	ユーザーまたはグループ
更新日時	日付と時刻
登録日時	日付と時刻
登録者	ユーザーまたはグループ
更新者	ユーザーまたはグループ

必須



カード

タイトル_DataCard1

ディスプレイ 詳細設定

Required

true

手間をかけないバリデーションチェック①

Requiredプロパティをtrueにする

カード

タイトル_DataCard1

ディスプレイ 詳細設定

Required

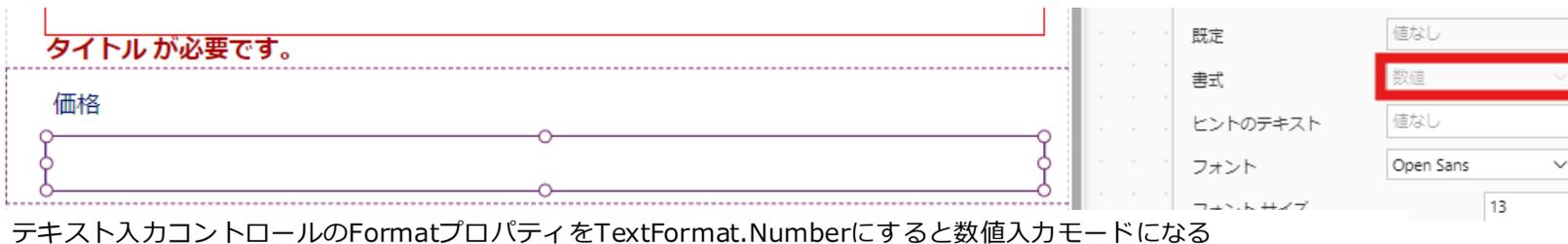
* タイトル

タイトルが必要です。

Updateプロパティが空白のときにエラーメッセージが表示される

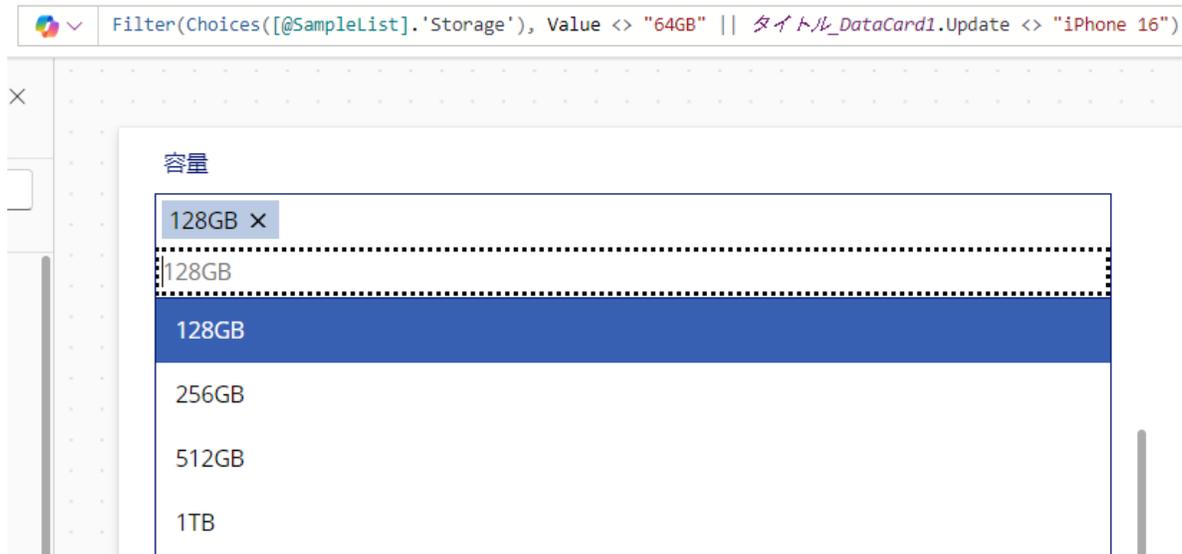
手間をかけないバリデーションチェック②

データ型に応じたコントロール選定・設定/プロパティの活用

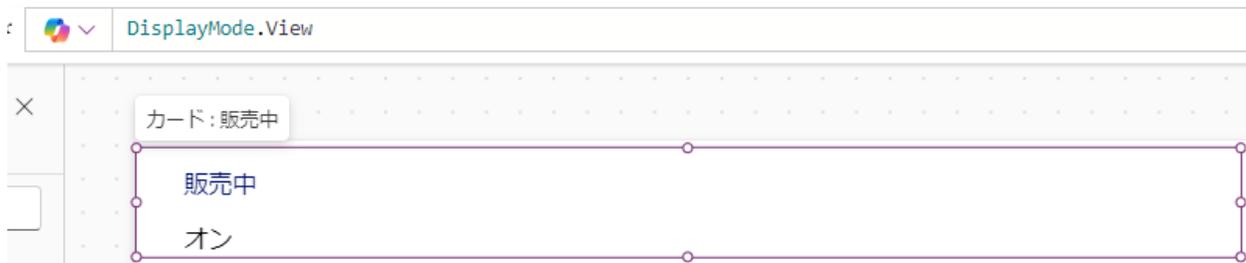


テキスト入力コントロールのFormatプロパティをTextFormat.Numberにすると数値入力モードになる

手間をかけないバリデーションチェック③ ユーザーに誤った値を選択させないロジック



特定の条件下では項目が表示されないようにフィルターをかける



他の項目から自動設定にしてカードをViewモードにする

バリデーションチェックの実装パターン を比較

エラーがあることをどのようにユーザーに知らせるか

	メリット	デメリット
1. ボタン押下時 エラーメッセージ表示	ユーザーが一通り入力してから エラーチェックができる	ボタンを押さないと エラーメッセージが表示されない
2. リアルタイムに エラーメッセージ表示	エラーの理由がすぐにわかる	ユーザーがストレスを感じる場合がある
3. ボタン非活性	正しく入力ができているかどうか 一目でわかる	どこでなぜエラーが起きているかが わかりにくい

エラーがあることをどのようにユーザーに知らせるか

	メリット	デメリット
1. ボタン押下時 エラーメッセージ表示	ユーザーが一通り入力してから エラーチェックができる	ボタンを押さないと エラーメッセージが表示されない
2. リアルタイムに エラーメッセージ表示	エラーの理由がすぐにわかる	ユーザーがストレスを感じる場合がある
3. ボタン非活性	バリデーションチェックではあまり使用しない	なぜエラーが起きているかが くい

バリデーションチェックの実装パターンを比較

	強み	弱み
Errorプロパティを利用したリアルタイムチェック(1+2)	複雑な数式や複数条件にも対応でき、 管理もしやすい エラーの内容を把握しやすい	工数がかかる 常にエラーメッセージが表示される ためユーザーがストレスを感じる
コントロールからフォーカスアウト時にチェック(2)	バリデーションチェックが複雑である場合、ユーザーストレスを軽減できる	工数がかかる 仕組み上多くの項目に対応するのは現実的でない
SubmitForm前にIfでチェック(1)	都度計算でないのでパフォーマンスが向上する(場合がある)	対象項目が多い場合数式が複雑化してしまう

要件ごとの推奨実装パターン

	バリデーションチェックの項目数	バリデーションチェックの条件
Errorプロパティを利用したリアルタイムチェック(1+2)	多い	複雑
コントロールからフォーカスアウト時にチェック(2)	少ない	複雑
SubmitForm前にIfでチェック(1)	少ない	簡単

Errorプロパティを利用した リアルタイムチェック

動作イメージ

The screenshot displays a product management interface. At the top, there is a header bar with a search field containing '商品名', a dropdown menu for '容量', and a sort order dropdown set to '商品名昇順'. On the right side of the header, there are buttons for '+ 追加' and '✓ 保存'.

The main content area is divided into two columns. The left column shows a list of iPhone models with their respective icons, prices, and screen sizes:

- iPhone 16 Pro: 229800, 6.3インチ
- iPhone 16: 124800, 6.1インチ
- iPhone SE (3rd Generation): 84800, 4.7インチ
- iPhone 14 Pro Max: 139800, 6.7インチ
- iPhone 16: 139800, 6.1インチ
- iPhone 16 Pro Max: 219800, 6.9インチ
- iPhone 16 Pro: (price and size not visible)
- iPhone 16 Pro: (price and size not visible)
- iPhone 16 Pro: (price and size not visible)
- iPhone 16 Pro: (price and size not visible)

The right column shows the details for the selected product, 'iPhone 16 Pro'. The fields are as follows:

- タイトル**: (empty)
- 価格**: | (input field)
- 画面サイズ**: 123 (input field)
- 発売日**: 1234 (input field)
- 発売日**: 1234456 (input field)
- 日付の選択...**: (calendar icon)
- スペック**: (empty)
- 容量**: 項目の検索 (dropdown menu)
- 販売中**: (toggle switch)
- 添付ファイル**: 添付されているものはありません。 (message box)
- 添付ファイル**: ファイルを添付 (button)

At the bottom of the interface, there is a pagination bar showing '26 - 50/94 (選択済み: 1)' and navigation arrows for '2 ページ'.

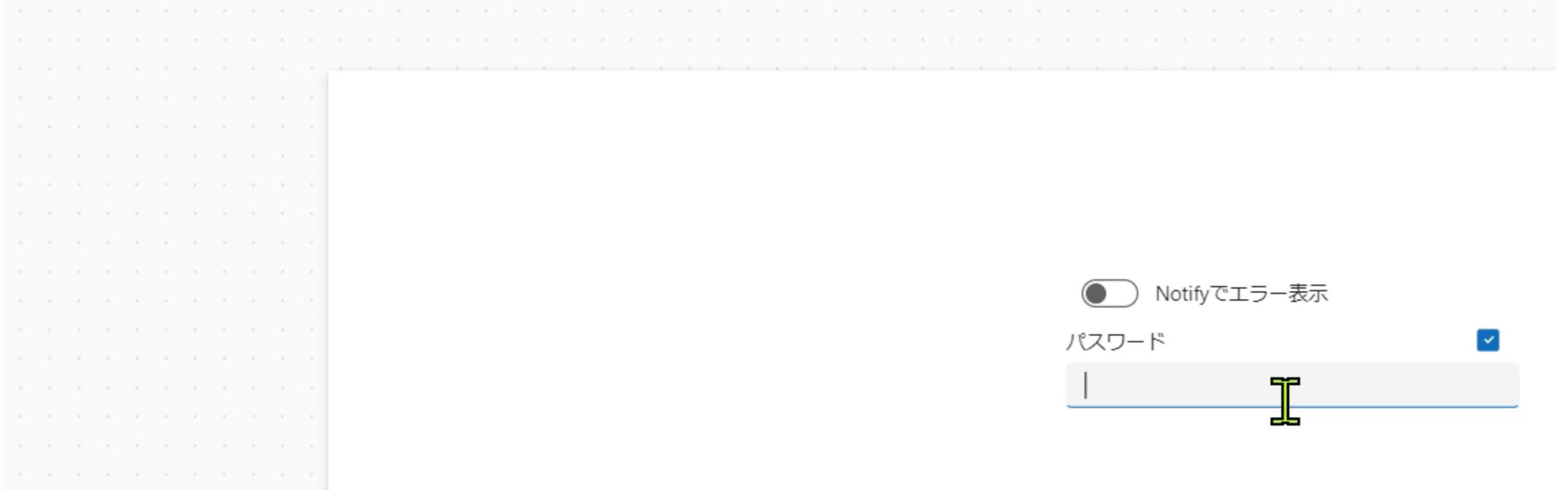
実装方法：複雑なのでアドベントカレンダーで投稿します

シリーズ1

日	月	火	水	木	金	土
1 @DEmodoriGatsuO なんか書く	2 @ayayan87360657 コーディング規約について書きます	3 @taku_maru 今年のキャンパスアプリの編集や作成周りであった更新情報のまとめ	4 @PowerPlatform4 エラーチェック方法色々 編集する	5 @pwrpla_tiger ユーザー定義関数のことを書きたい	6 @ksgiksg なにか書きます	7 @youseibubu なんか書きます！
8 @Yo_8One 複雑な条件にする際のIF関数の書き方を書こうかな	9 @DEmodoriGatsuO 何か書きます	10 @AkioSaitoh 2024年 Power Apps Weekly News 学び3行ポスト	11 @PowerPlatform4 複雑なエラーチェックが必要なおすすめ実装 編集する	12 @tsukapon 具のリスキリング講座のローコード開発基礎 (Power Apps)を受けてみて	13 @f-sumiko 何か書きます	14 @youseibubu なんか書きます！2
15 @dai365 書きます	16 @DEmodoriGatsuO 何か書きます	17 @aoi2021 何か書きます	18 @PowerPlatform4 Outlookメールからテキスト抽出&フォルダ移動の方法 編集する	19 @susumutanaka なんか書きます	20 @go_bitzemi リクエストあった画面作成の手順を書いて個人ブログ始めます。	21 @yuRi_67lily 何か書きます！
22 @dai365 書きます	23 @Masayanf 何か書きます	24 @DEmodoriGatsuO 何か書きます	25 @PowerPlatform4 Dataverseのテーブル情報を取得する方法(仮) 編集する	26	27	28

**コントロールからフォーカス
アウト時にチェック**

動作イメージ



Notifyでエラー表示

パスワード

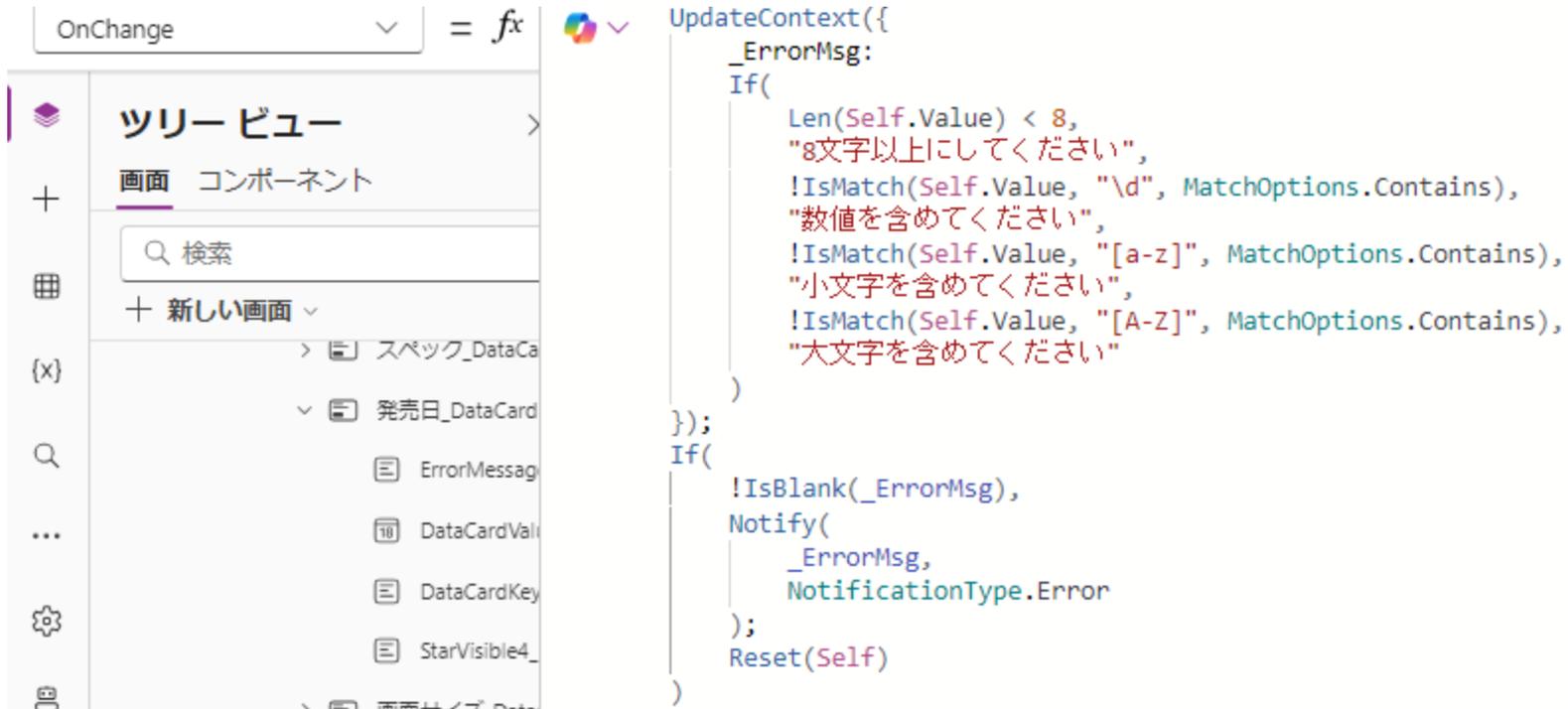


| I

実装方法(Notify関数でエラーメッセージ表示)

テキスト入力

OnChange = fx



```
UpdateContext({
  _ErrorMsg:
  If(
    Len(Self.Value) < 8,
    "8文字以上にしてください",
    !IsMatch(Self.Value, "\d", MatchOptions.Contains),
    "数値を含めてください",
    !IsMatch(Self.Value, "[a-z]", MatchOptions.Contains),
    "小文字を含めてください",
    !IsMatch(Self.Value, "[A-Z]", MatchOptions.Contains),
    "大文字を含めてください"
  )
});
If(
  !IsBlank(_ErrorMsg),
  Notify(
    _ErrorMsg,
    NotificationType.Error
  );
Reset(Self)
)
```

実装方法(ラベルにエラーメッセージ表示)

テキスト入力

OnChange = fx

ツリービュー
画面 コンポーネント
検索
新しい画面
スペック_DataCa
発売日_DataCard
ErrorMessage
DataCardVal
DataCardKey

```
UpdateContext({
  _ErrorMsg:
  If(
    Len(Self.Value) < 8,
    "8文字以上にしてください",
    !IsMatch(Self.Value, "\d", MatchOptions.Contains),
    "数値を含めてください",
    !IsMatch(Self.Value, "[a-z]", MatchOptions.Contains),
    "小文字を含めてください",
    !IsMatch(Self.Value, "[A-Z]", MatchOptions.Contains),
    "大文字を含めてください"
  )
});
If(
  !IsBlank(_ErrorMsg),
  Reset(Self)
);
```

エラー表示ラベル

Text = fx

_ErrorMsg

_ErrorMsg = 空白 データ型: テキスト

SubmitForm前にIfでチェック

動作イメージ

The screenshot displays a mobile application interface for managing product listings. At the top, there is a blue header bar with a search bar containing '商品名' and '容量', and buttons for '+ 追加' and '✓ 保存'. Below the header, a list of iPhone models is shown on the left, each with a circular icon containing 'i1', the model name, price, and screen size. The selected item is 'iPhone 16 Pro' with a price of 154800 and a 6.3-inch screen. To the right of the list is a detailed edit form for the selected item, titled 'タイトル'. The form includes fields for '価格', '画面サイズ', '発売日', 'スペック', '容量', and '販売中'. The '販売中' field is a toggle switch that is currently turned on. Below these fields is a section for '添付ファイル' (Attachments), which contains a message: '添付されているものはありません。' and a button labeled '📎 ファイルを添付'. At the bottom of the list, there is a pagination indicator: '1 - 25/95 (選択済み: 1)' and navigation arrows.

商品名	容量
iPhone 16 Pro	154800 6.3インチ
iPhone 15	129800 6.1インチ
iPhone 16 Plus	189800 6.7インチ
iPhone 15	159800 6.1インチ
iPhone 15 Pro	129800 6.1インチ
iPhone 14 Pro	119800 6.1インチ
iPhone 16 Pro	199800 6.3インチ
iPhone 15 Plus	149800 6.7インチ
iPhone 16 Pro Max	174800 6.9インチ
iPhone 16 Pro	154800

1 - 25/95 (選択済み: 1) 1 ページ

実装方法

保存ボタン

```
OnSelect = fx  
If(  
    発売日_DataCard1_1.Update > Today(),  
    Notify("今日以前の日付を入力してください。", NotificationType.Error),  
    SubmitForm(Form1_1)  
)
```

**複雑なバリデーションチェックでは
入れ子になってしまうので
この方法は使用しない！！！！！！！！**

【Appendix】 バリデーションチェックでよく使う関数

関数

Notify関数：エラーメッセージなどを画面上部に数秒間通知する関数

```
Notify(  
    _ErrorMsg,  
    NotificationType.Error  
);
```

IsBlank関数：必須チェックや入力時のみエラーチェックする条件を作る関数

```
If(  
    And(  
        Value(DataCardValue2.Value) < 100000,  
        !IsBlank(DataCardValue2.Value)  
    ),  
    "価格は100000以上にしてください。"
```

IsMatch関数：複雑な条件を正規表現でチェックする関数

```
!IsMatch(Self.Value, "\d", MatchOptions.Contains),  
"数値を含めてください",  
!IsMatch(Self.Value, "[a-z]", MatchOptions.Contains),  
"小文字を含めてください",  
!IsMatch(Self.Value, "[A-Z]", MatchOptions.Contains),  
"大文字を含めてください"
```

Len関数：複雑な条件を正規表現でチェックする関数

```
If(  
    Len(Self.Value) < 8,  
    "8文字以上にしてください",
```

プロパティ

DataCard.Required : trueのときにUpdateプロパティが空白だとエラー

```
Required = fx !IsBlank(ErrorMessage2.Text)
```

DataCard.Error : Requiredでエラーになったときに〇〇が必要ですという文字列を出力する

```
Required = fx !IsBlank(ErrorMessage2.Text)
```

DataCard.Valid : IsBlank(DataCard.Error)

```
Required = fx !IsBlank(ErrorMessage2.Text)
```

Form.OnFailure : SubmitFormが失敗した時に実行される処理

```
OnFailure = fx Notify(LookUp(_errors, !IsBlank(msg), msg), NotificationType.Error)
```

Form.Valid : フォームにエラーがあるときにfalseになる

```
If(
  Form1.Valid,
  DisplayMode.Edit,
  DisplayMode.Disabled
)
```

Form.Error/ErrorKind : Errors関数の出力とほぼ同じ

ありがとうございました