【Power Appsでハッカソンしたいけぇ#10】

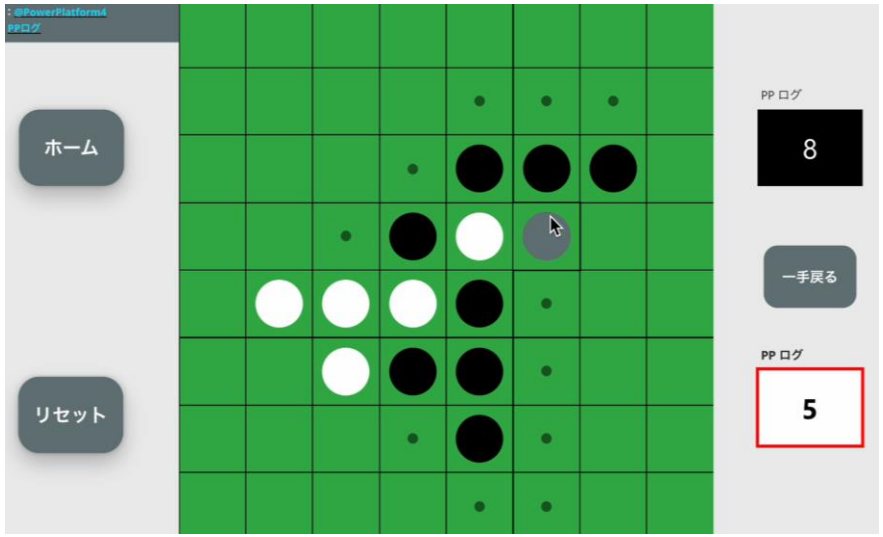
Power Appsの
バリデーションチェック(エラーチェック)

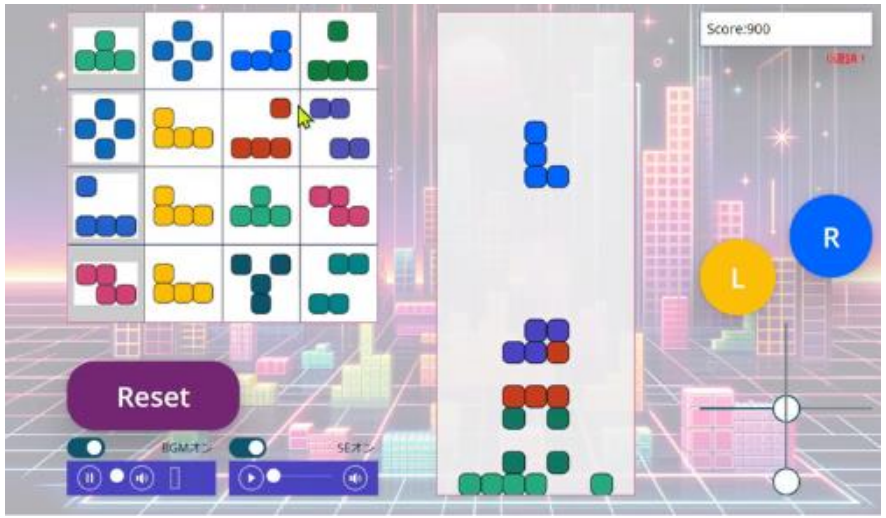
～PPの城窓から～

PPログ 2024/11/24

PPログ

X(Twitter)：
https://twitter.com/PowerPlatform4
Blog：
https://powerplatformnikki.com/





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

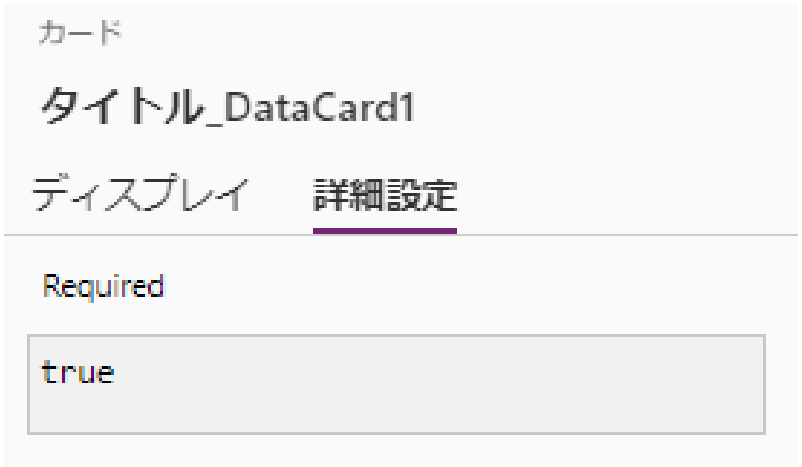
まえおき

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

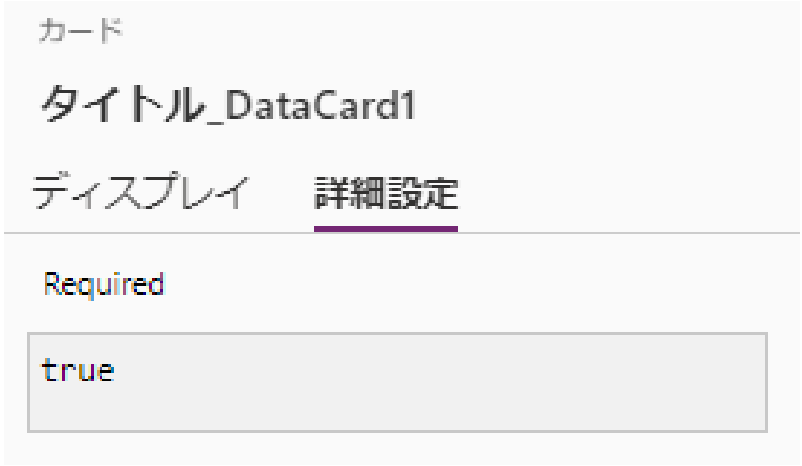
列

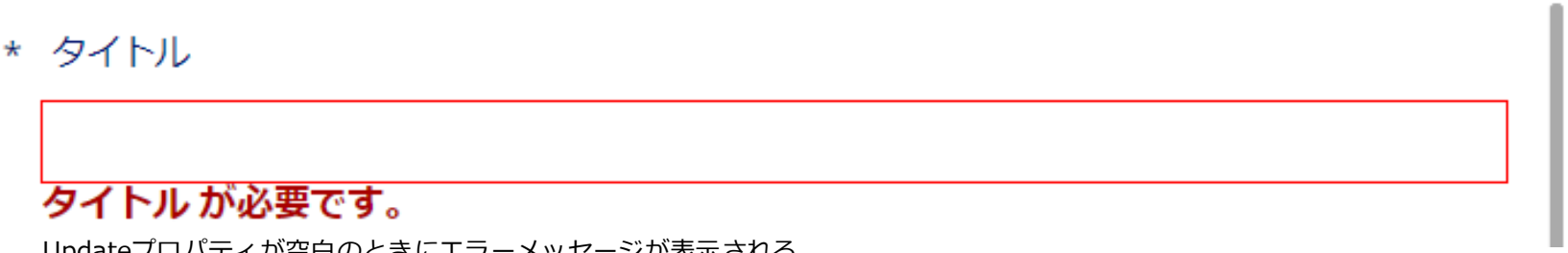
列には、リスト内の各アイテムについての情報が保存されます。現在、このリストでは次の列を使用できます。

列 (クリックして編集)	種類	必須
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	



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



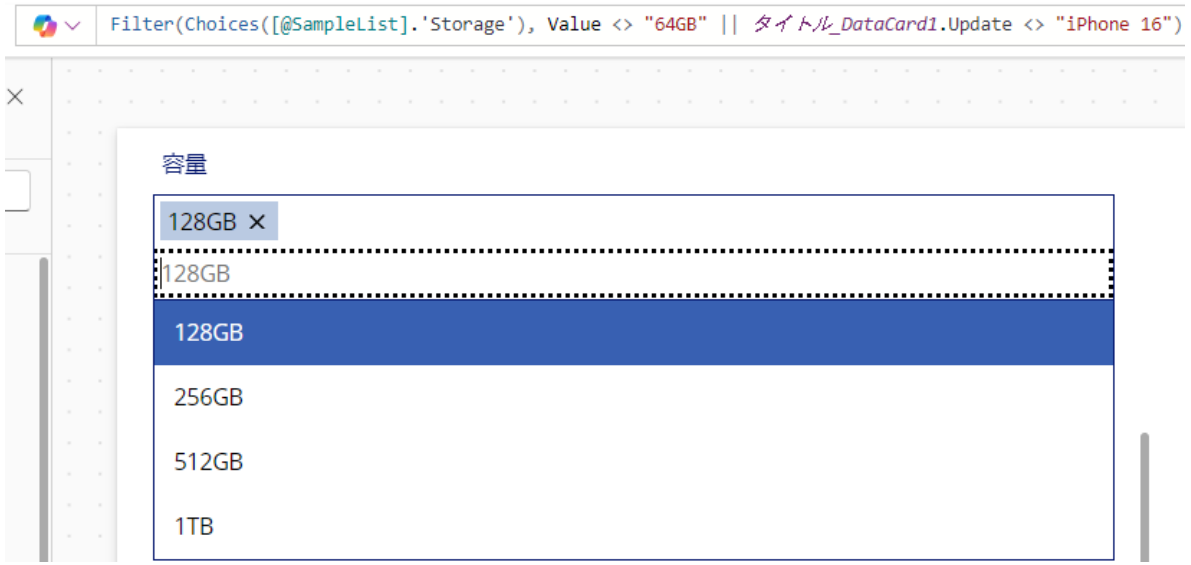


Updateプロパティが空白のときにエラーメッセージが表示される

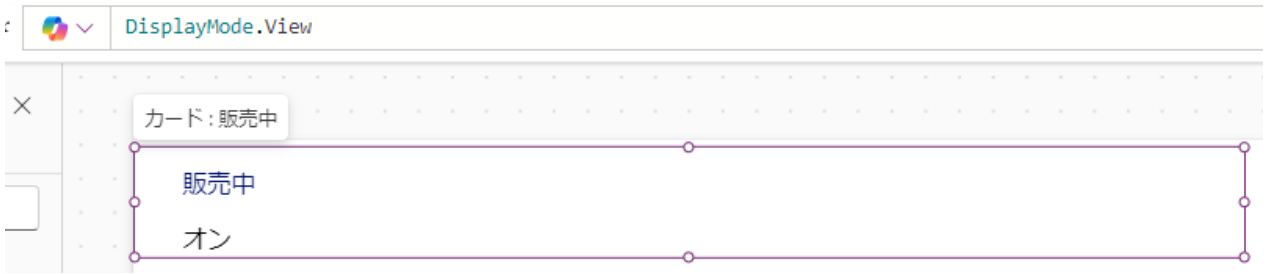
手間をかけないバリデーションチェック②
データ型に応じたコントロール選定・設定/プロパティの活用

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が 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**	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

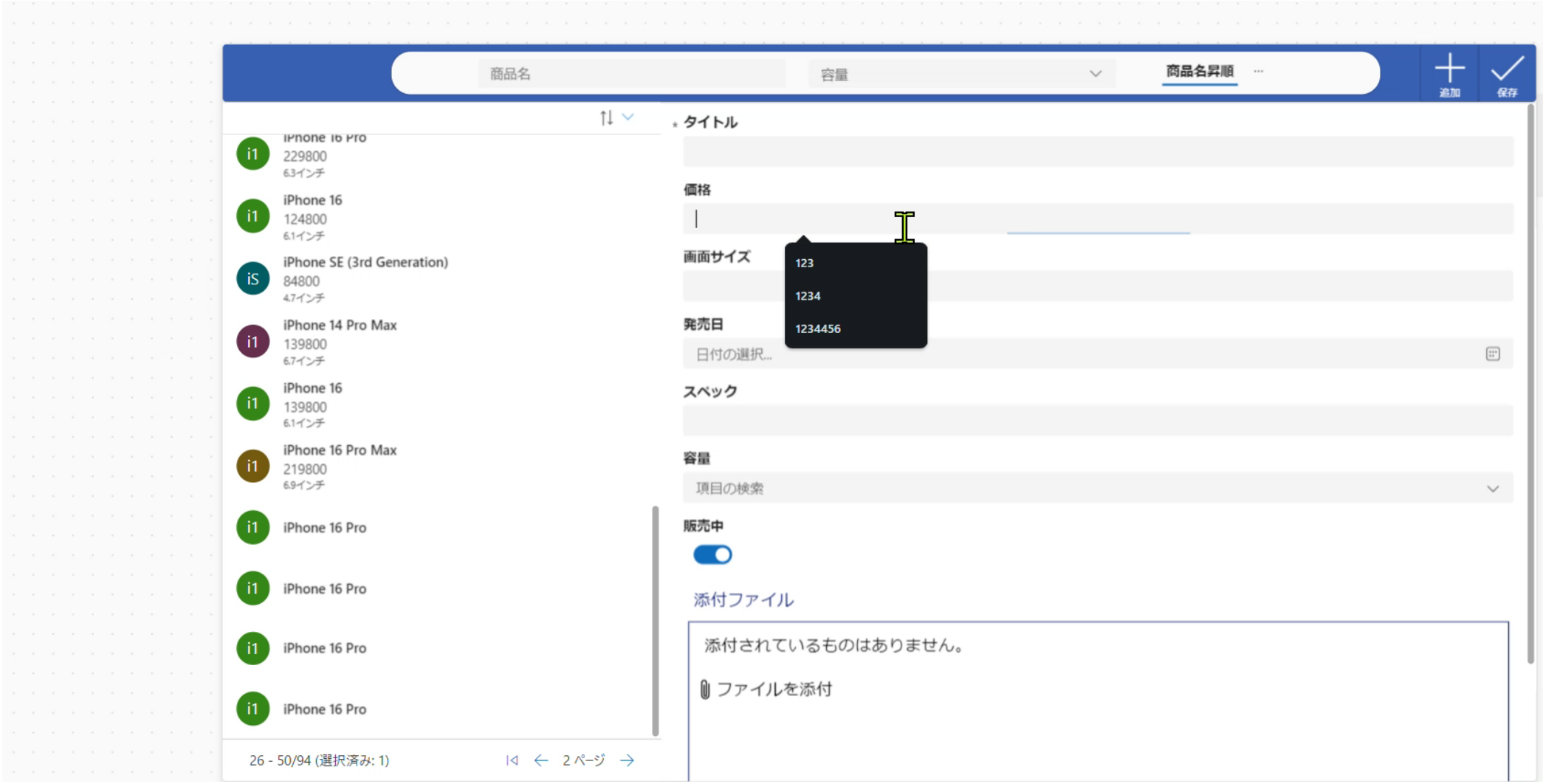
	強み	弱み
Errorプロパティを利用した リアルタイムチェック(1+2)	複雑な数式や複数条件にも対応でき、 管理もしやすい エラーの内容を把握しやすい	工数がかかる 常にエラーメッセージが表示される ためユーザーがストレスを感じる
コントロールからフォーカス アウト時にチェック(2)	バリデーションチェックが複雑で ある場合、ユーザーストレスを 軽減できる	工数がかかる 仕組み上多くの項目に対応するのは 現実的でない
SubmitForm前にIfで チェック(1)	都度計算でないのでパフォーマンス が向上する(場合がある)	対象項目が多い場合数式が複雑化 してしまう

要件ごとの推奨実装パターン

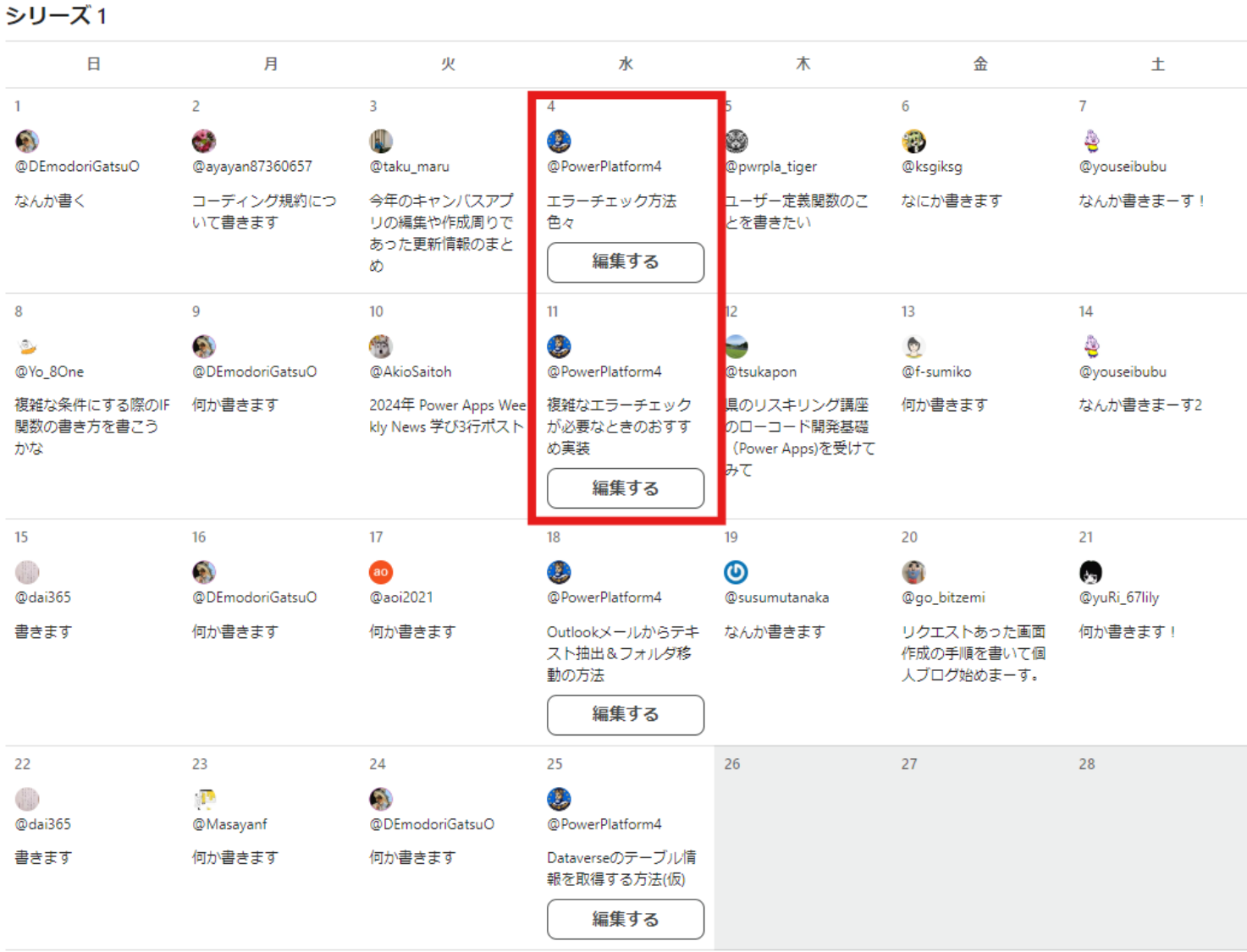
	バリデーションチェックの項目数	バリデーションチェックの条件
Errorプロパティを利用したリアルタイムチェック(1+2)	多い	複雑
コントロールからフォーカスアウト時にチェック(2)	少ない	複雑
SubmitForm前にIfでチェック(1)	少ない	簡単

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

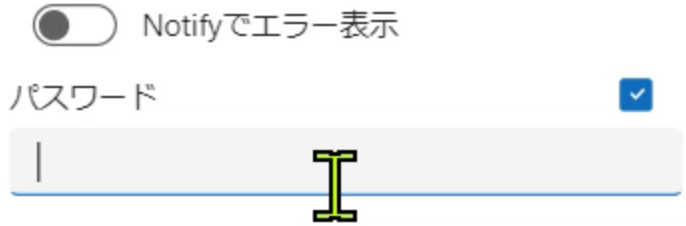

商品名
容量
商品名昇順
追加
保存
iPhone 16 Pro
229800
6.3インチ
iPhone 16
124800
6.1インチ
iPhone SE (3rd Generation)
84800
4.7インチ
iPhone 14 Pro Max
139800
6.7インチ
iPhone 16
139800
6.1インチ
iPhone 16 Pro Max
219800
6.9インチ
iPhone 16 Pro
iPhone 16 Pro
iPhone 16 Pro
iPhone 16 Pro
26 - 50/94 (選択済み: 1)
2 ページ
タイトル
価格
123
1234
1234456
画面サイズ
発売日
日付の選択...
スペック
容量
項目の検索
販売中
添付ファイル
添付されているものはありません。
ファイルを添付

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



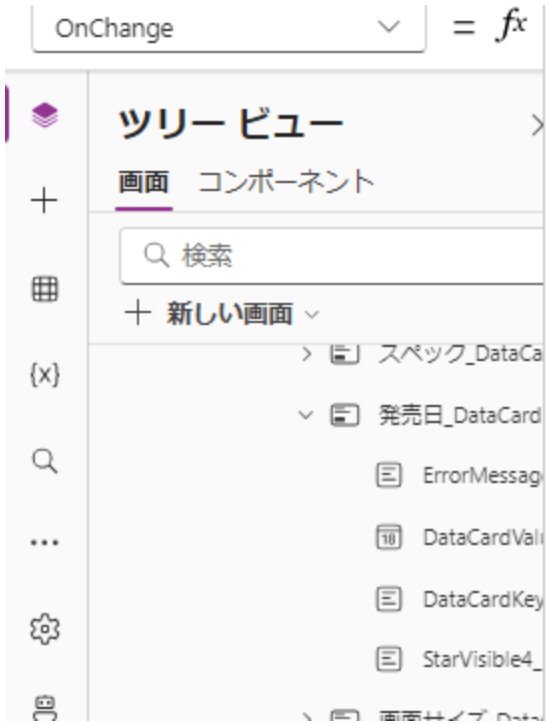
コントロールからフォーカスアウト時にチェック

動作イメージ



実装方法(Notify関数でエラーメッセージ表示)

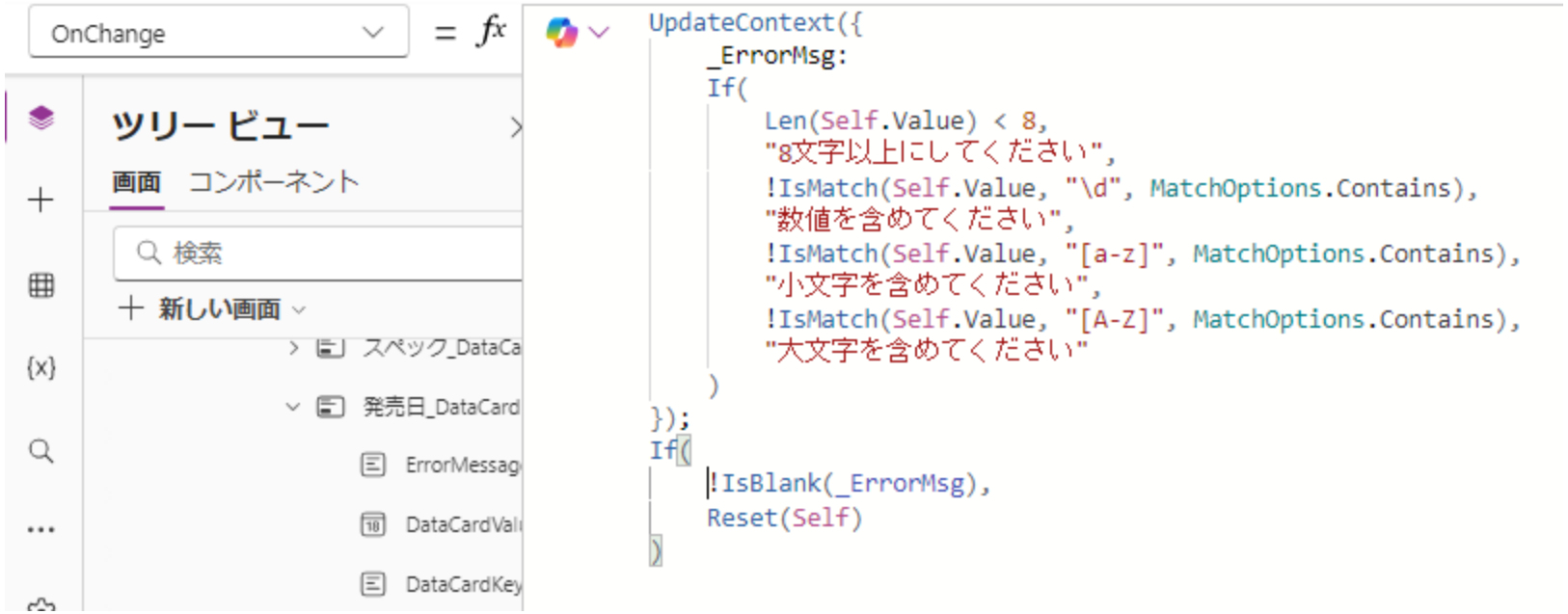
テキスト入力



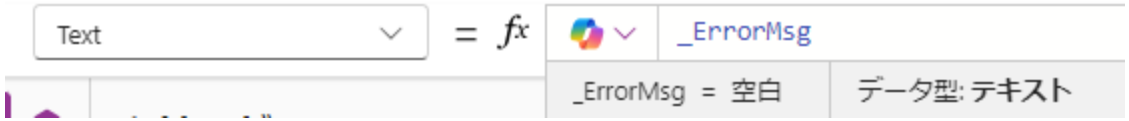

```
UpdateContext({
    _ErrorMsg:
    If(
        Len(Self.Value) < 8,
        "8文字以上にしてください",
        !IsMatch(Self.Value, "\d", MatchOptions.Contains),
        "数値を含めてください",
        !IsMatch(Self.Value, "[a-z]", MatchOptions.Contains),
        "小文字を含めてください",
        !IsMatch(Self.Value, "[A-Z]", MatchOptions.Contains),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



エラー表示ラベル



SubmitForm前にIfでチェック

動作イメージ


商品名
容量
追加
保存
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
タイトル
tes
価格
画面サイズ
発売日
日付の選択...
スペック
容量
項目の検索
販売中
添付ファイル
添付されているものはありません。
ファイルを添付

実装方法

保存ボタン

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

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

DataCard.Valid： IsBlank(DataCard.Error)

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

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