HCL Volt MX Go
VMXGo-DEV-100

Lesson 6 – セグメントのソートと フィルタリング

もくじ

はじめに

HCL Volt MX Go VMXGo-DEV-100 トレーニングコースは、現在のリリースである Volt MX Go v2.0.1 (https://opensource.hcltechsw.com/voltmxgo-documentation/references/whatisnew.html) 以降の HCL Volt MX Go ツールを学ぶための開発者向けスタートコースです。

Volt MX Go の開発者が知っておかなければならない 2 つの主要なツールは、1) Design Import と 2) VoltFormula です。Design Import は、Volt MX Go Iris（別名 Volt Iris）のプロジェクト、UI、Domino とデータを交換するロジックをゼロから作成する必要がなく、開発者の時間と労力を大幅に削減します。VoltFormula を使用すると、依存する Domino ロジックを HCL Volt MX プロジェクトで使用できるため、JavaScript でロジックを書き直す時間と労力を節約できます。

Design Import は、Domino バージョン 12.0.2 以降でホストされている HCL Domino アプリケーション（Domino REST API、別名 DRAPI に公開済み）を Volt MX Go の Volt Iris プロジェクトと Volt MX Go の Volt Foundry アプリ（Foundry ミドルウェアサービスのコレクション）にインポートします。Design Import の最終的な出力は、Volt Iris Web アプリで、すぐに機能し、完全に開発され、Domino アプリケーションを表すすべての Volt Iris フォームとウィジェットが含まれ、OAuth2 Identity サービス、Integration サービス、Volt MX Go の Foundry Domino アダプターを使用する Object サービスを持つ Foundry アプリに関連付けられています。

Design Import 後における Volt MX Go アプリ開発の一般的な流れは、組織のブランディングや UI 要件に合わせて UI のリブランド/リファクタリングを行い、Domino 文書のリストにソートやフィルタリング機能を追加することです。

この HCL Volt MX Go VMXGo-DEV-100 トレーニングには、上記を扱う 6 つのレッスンが含まれています。レッスンは以下の通りです。

1. Lesson 1 - Domino REST API 必須情報
2. Lesson 2 - Design Import のセットアップ
3. Lesson 3 - Design のインポート
4. Lesson 4 - VoltFormula
5. Lesson 5 - UI のリブランディング
6. Lesson 6 - セグメントのソートとフィルタリング

このコースでは、HCL Volt MX Go First Touch Recipe Catalog アプリとその資産（Domino DB（レシピ保存用）、First Touch Recipe Domino REST API スキーマ、スコープ、DRAPI アプリ

（https://opensource.hcltechsw.com/voltmxgo-documentation/tutorials/firsttouch.html）を含む）を活用します。DRAPI First Touch Recipe アプリで Design Import を実行し、Volt Iris アプリに VoltFormula を追加し、Iris アプリのログイン画面とダッシュボード画面/フォームをリブラン ドし、Iris アプリにソートとフィルタリング機能を追加します。

前提条件

このコースを修了するには、HCL Volt MX Go の Volt Foundry（ミドルウェア）と Volt Iris（IDE）に加え、Domino REST API を含む Domino 環境が必要です。Domino と Volt MX Go サーバーのオンプレミスインストールの代わりに、HCL SoFy プラットフォーム（https://hclsofy.com）の HCL Volt MX Go サンドボックスを使用できます。HCL SoFy サンドボックスには、Domino、Domino REST API、および Volt MX Go Foundry が含まれます。SoFy が提供するトライアルサンドボックスを使用するには、付録 I を参照してください。

オンプレミス

- HCL Domino server 12.0.2+
- HCL Domino REST API (DRAPI) サービス (タスクとサービスが稼動)
- HCL Domino REST API Console URL
- HCL Domino REST API Admin User Credentials (ユーザーID とパスワード)
- HCL Volt MX Go Foundry v2.0.1
- HCL Volt MX Go Foundry Console URL
- HCL Volt MX Go Foundry Admin User Credentials (ユーザーID とパスワード)
- HCL Volt MX Go Iris v2.0.1

HCL SoFy プラットフォーム（https://hclsofy.com）

- HCL Volt MX Go サンドボックス

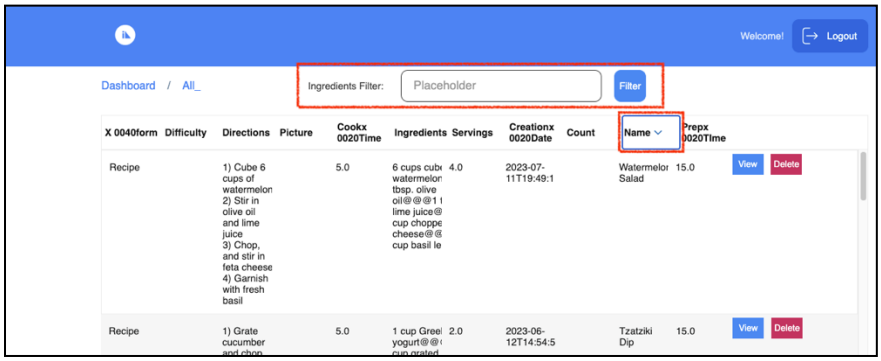
コピー可能なコードソース

> 注意：設定やコードスニペットを Volt Foundry や Iris にコピーするよう求められたら、**Lesson_6-Segment_Sorting&Filtering-Copiable_Source.txt** ファイルからコピーしてください。このドキュメントからコピーすると、不要な制御文字や隠された制御文字が含まれている可能性があり、Foundry サービスや Iris アプリが正しく動作しなくなります。

Lesson 6 – セグメントのソートとフィルタリング

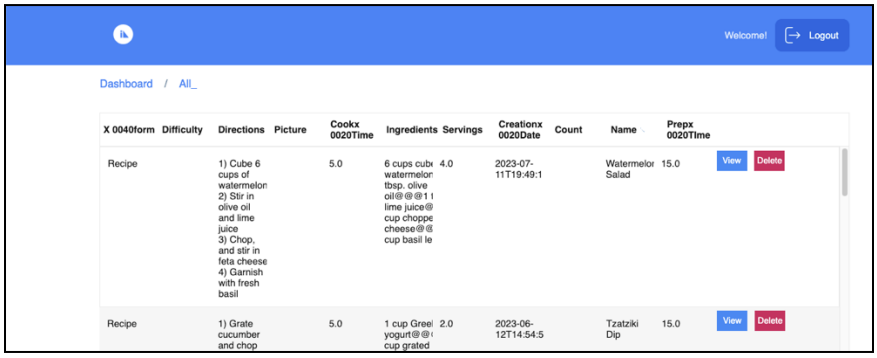
このレッスンでは、レシピのソートとフィルタリングを実装する方法を説明します。リストのソートとフィルタリングを実現するには、Volt Foundry Object サービス呼び出しに "OData" クエリを適用して目的のレコードを取得するなど、さまざまな方法があります。しかし、保持しているレコードの量が限られており、ネットワーク呼び出しを避けたい場合は、このレッスンのメソッドを使用してデータレコードをソートまたはフィルタリングできます。

まず、レシピのリストを名前カラムでソートします。このレッスンでは、昇順と降順の並べ替え方法を説明します。次に、食材（例えばレモン）を入力するためのテキストボックスを作成し、指定した食材を含むレシピを絞り込む方法を説明します。



Recipe Sorting の実装

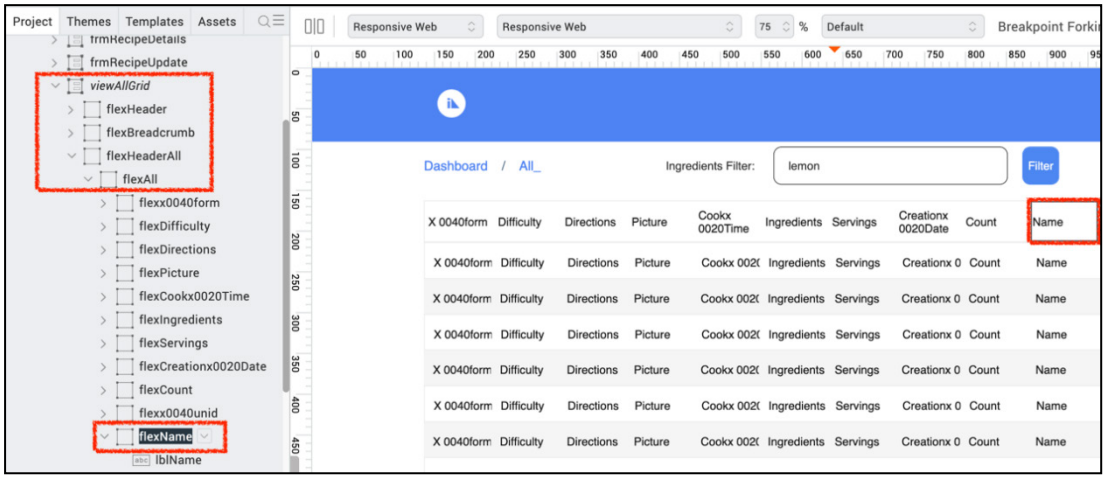
Volt Iris Recipes プロジェクト (Volt MX Go Design Import で作成) で一覧表示されているレシピを viewAllGrid フォームでソートする機能を実装します。ログイン後、ダッシュボードの Views -> All をクリックするとフォーム viewAllGrid に Domino レシピデータベースのすべてのレシピが表示されます。



セグメントのソートとフィルタリング

ステップ

1. Volt Go Iris IDE から、IDE が前のレッスンで Design Import によって作成されたレシピ プロジェクトですでに開かれていることを確認します。Iris キャンバスの Design ビューにいることを確認します（Storyboard ビューではありません）。
2. IDE の左上側で Project タブの下のフォームツリー構造をナビゲートして、フォーム viewAllGrid を開きます。フォームが Iris センターキャンバスに表示されているはずです。
3. flexName ウィジェットの隣に Name のソート方向を示す画像を追加し、ソートコードを追加し、ユーザーがクリックしたときにソートを適用するロジックを追加します。
   - ウィジェットのツリー パス viewAllGrid -> flexHeaderAll -> flexAll にあるウィジェット flexName をクリックして Iris にフォーカスを設定します。
   - PROPERTIES -> FlexContainer -> Layout Type に移動します。



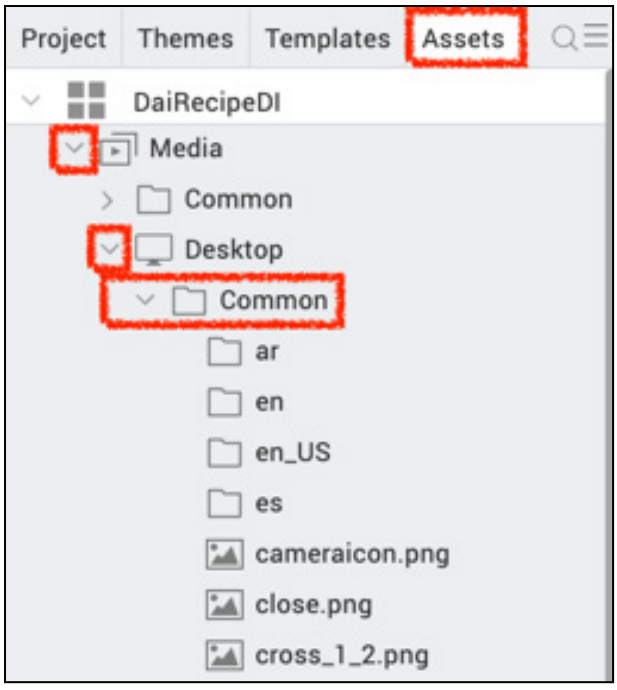
   - Layout Type を Flow Horizontal に設定します。
   - flexName の子ウィジェット lblName をクリックして Iris にフォーカスを設定します。
   - PROPERTIES -> Look -> Width に移動します。
   - lblName の Width プロパティを Preferred に変更します。
   - Iris フォーカスを flexName に設定します。
   - Default Library から Image ウィジェットを中央のキャンバスにドラッグし、ウィジェット flexName（テキストラベルは "Name"）にドロップします。

セグメントのソートとフィルタリング

- 新しい Image ウィジェットを右クリックし、名前を imgSortDir に変更します。
- PROPERTIES -> Look に移動し、以下のプロパティを設定します。
    i. Left: 0 Dp,
    ii. Top: 0 Dp,
    iii. Width: Preferred,
    iv. Height: 80%,
    v. Center: 50%.
- Iris プロジェクトを保存します。
- このドキュメントにある以下の 2 つのアイコンをダウンロードフォルダに保存し、表示されているとおりの名前を付けてください。

 imgchevrondown.png  imgchevronup.png

- Project タブ -> Assets -> Media -> Desktop -> Common に移動します。



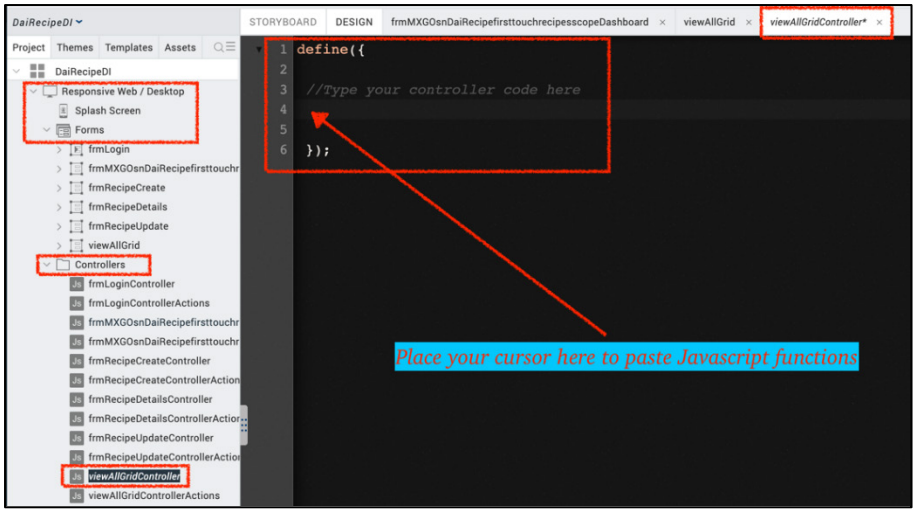
- Right-click Desktop Common to Import Images,

- Desktop Common で右クリックして Import Images を選択します。
- ポップアップウィンドウが表示されます。ダウンロードフォルダから imgchevrondown.png と imgchevronup.png の 2 つの画像をインポート／アップロードします。
- トップレベルのメニュー Project -> Refresh をクリックして、 Iris プロジェクトに Assets 内の新しい 2 つの画像が表示されるようにします。
- Iris のフォーカスをウィジェット imgSortDir に設定します。
- PROPERTIES -> Image -> Source -> Edit に移動します。
- imgchevrondown.png が表示されるまでポップアップウィンドウをスクロールし、それを選択して OK をクリックします。
- 下向き矢印アイコンがレシピリストのヘッダー行の Name カラムの横に表示されます。

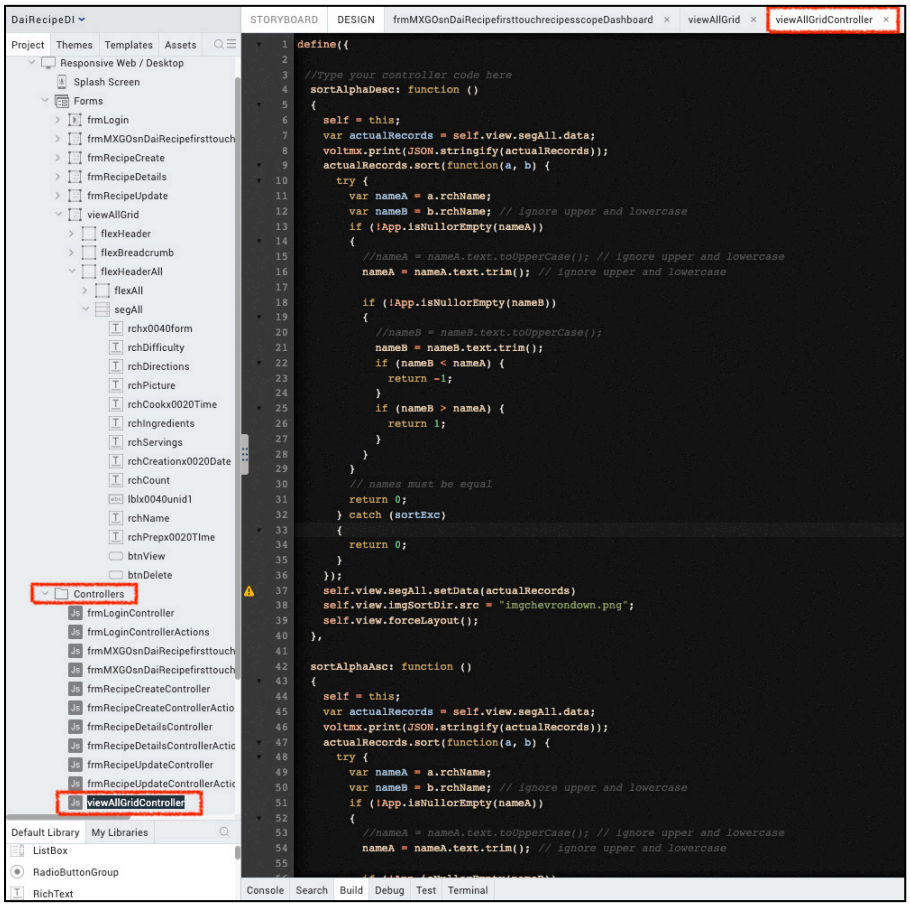
フォーム viewAllGrid のコントローラモジュール viewAllGridController に 2 つの Javascript (JS)関数を追加します。

- Project -> Responsive Web / Desktop -> Controllers -> viewAllGridController に移動します。
- viewAllGridController をクリックして、中央のキャンバスで開きます。コントローラは最上部と最下部の行を除いてほとんど空になります。

セグメントのソートとフィルタリング



- 次のページのテキストボックスから関数 **sortAlphaDesc** と **sortAlphaAsc** をコピーし、現在 **viewAllGridController** Javascript モジュールを表示しているエディタの中央行に貼り付けます（エディタ画面の例を以下に示します）、



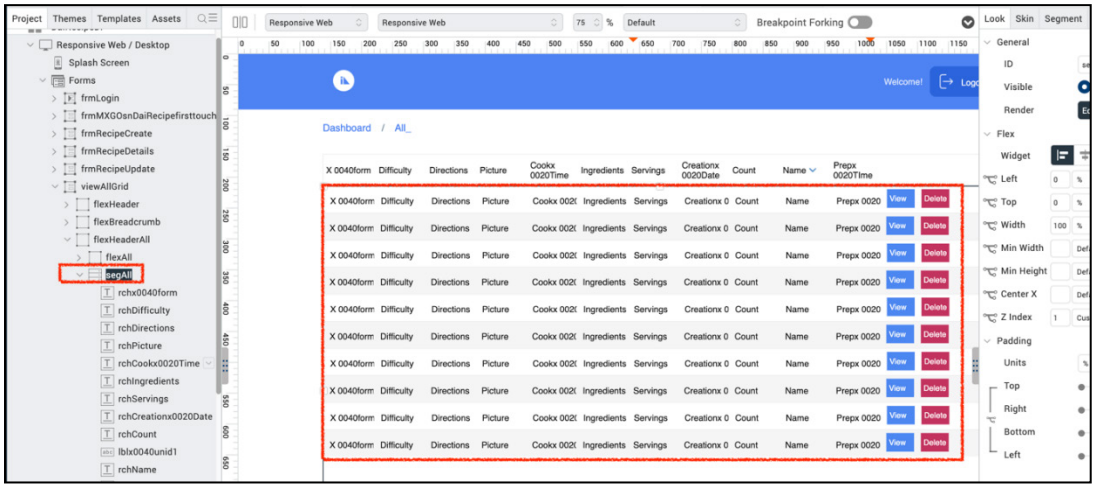
セグメントのソートとフィルタリング

```
sortAlphaDesc: function ()
{
  self = this;
  var actualRecords = self.view.segAll.data;
  voltmx.print(JSON.stringify(actualRecords));
  actualRecords.sort(function(a, b) {
    try {
      var nameA = a.rchName;
      var nameB = b.rchName; // ignore upper and lowercase
      if (!App.isNullorEmpty(nameA))
      {
        nameA = nameA.text.trim(); // ignore upper and lowercase

        if (!App.isNullorEmpty(nameB))
        {
          nameB = nameB.text.trim();
          if (nameB < nameA) {
            return -1;
          }
          if (nameB > nameA) {
            return 1;
          }
        }
      }
      // names must be equal
      return 0;
    } catch (sortExc)
    {
      return 0;
    }
  });
  self.view.segAll.setData(actualRecords)
  self.view.imgSortDir.src = "imgchevrondown.png";
  self.view.forceLayout();
},
```

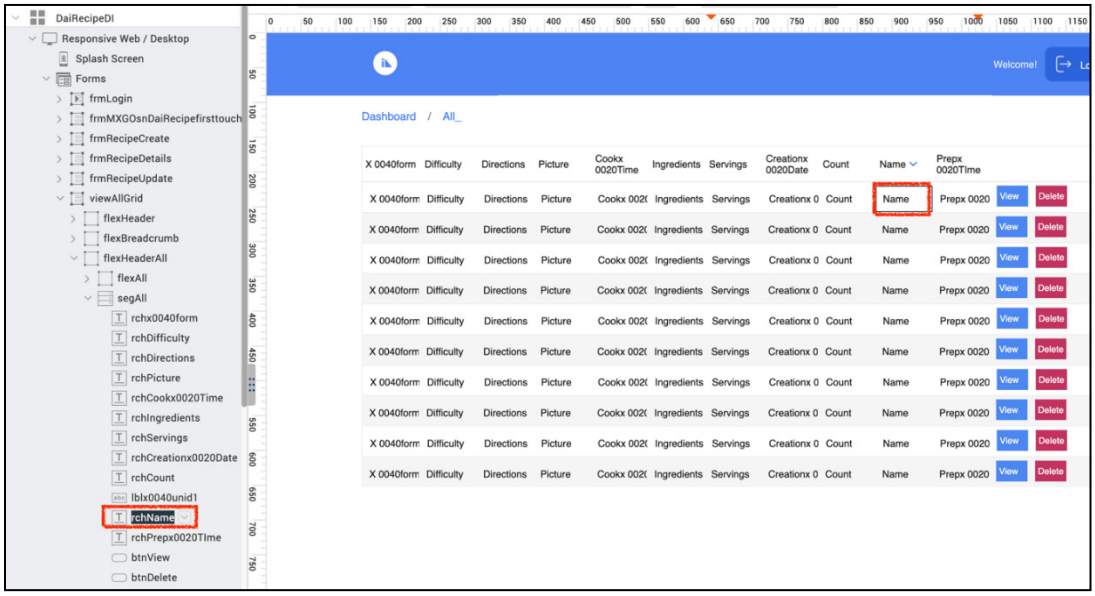
```
sortAlphaAsc: function ()
{
  self = this;
  var actualRecords = self.view.segAll.data;
  voltmx.print(JSON.stringify(actualRecords));
  actualRecords.sort(function(a, b) {
    try {
      var nameA = a.rchName;
      var nameB = b.rchName; // ignore upper and lowercase
      if (!App.isNullorEmpty(nameA))
      {
        nameA = nameA.text.trim(); // ignore upper and lowercase

        if (!App.isNullorEmpty(nameB))
        {
          nameB = nameB.text.trim();
          if (nameA < nameB) {
            return -1;
          }
          if (nameA > nameB) {
            return 1;
          }
        }
      }
      // names must be equal
```

セグメントのソートとフィルタリング

注：2 つの JS 関数のうち、**self.view.segAll** はセグメントウィジェットで、Iris プロジェクトで **viewAllGrid** フォームのすべての既存レシピをリストするために使用されます。セグメントウィジェットは非常に堅牢で強力なウィジェットで、ウィジェットのリストを豊富に表示できます。このセグメントとそのコンテンツはすべて Design Import によって作成されるため、一から作成する手間が省けます。セグメント名が異なる場合は、2 つの関数でセグメント名を変更してください。



注 2: JS コードにある **rchName** はリッチテキストウィジェットです。これはレシピ名を保持し、**Design Import** によって作成された Segment に存在している必要があります。ユーザーがヘッダー行の **Name**（矢印アイコン）をクリックしたときにレシピ名でソートを実装しています。

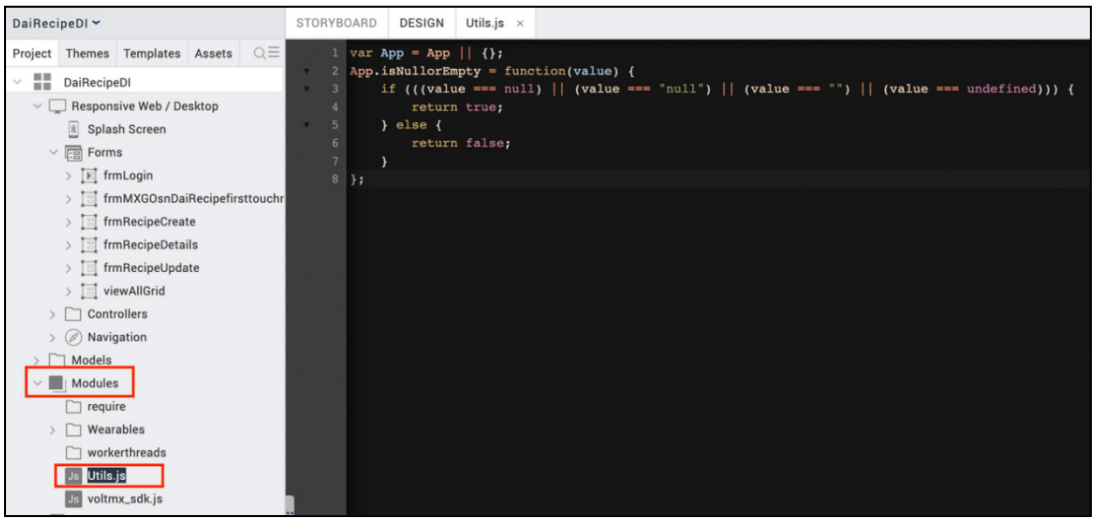


- Iris プロジェクトを保存します。

- 上記の 2 つの JS 関数は、共通のユーティリティ関数 App.isNullorEmpty()を使用しています。これを共通のユーティリティ JS Module に追加します。

```
var App = App || {};
App.isNullorEmpty = function(value) {
    if (((value === null) || (value === "null") || (value === 
"") || (value === undefined))) {
        return true;
    } else {
        return false;
    }
};
```

- Project -> Modules に移動します。
- Modules を右クリックして New JS Module を選択します。
- 新しい JS モジュールをクリックして Iris にフォーカスを設定します。
- 右クリックして、新しいモジュールの名前を Utils.js に変更します。
- 上のテキストボックスから関数 App.isNullorEmpty()をコピーし、中央のキャンバスに表示されている Utils.js エディタに貼り付けます。

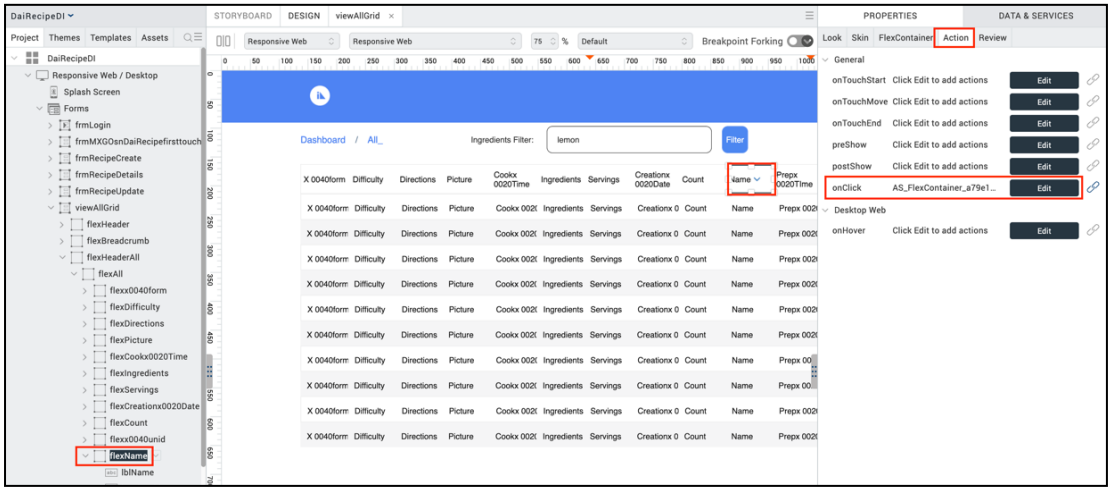


- Iris プロジェクトを保存します。

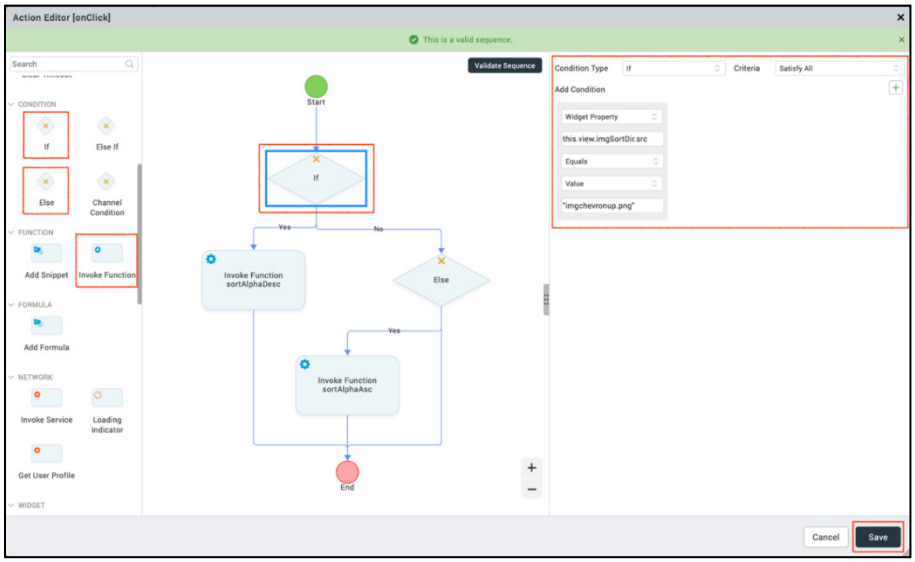
ウィジェット flexName に onClick イベントを追加して、 ユーザーがクリックするとレシピのリストがソートされるようにします。ソートの方向は flexName の隣に表示される上か下のアイコンによって決まります。ローコードのアクションスクリプトを使って、前回追加した 2 つのソート関数のうち 1 つを呼び出します。

セグメントのソートとフィルタリング

- Project -> Responsive Web / Desktop -> Forms -> viewAllGrid -> flexHeaderAll -> flexAll -> flexName のパスから Iris のフォーカスを flexName に設定します。
- PROPERTIES -> Action -> onClick イベントに移動します。
- Edit をクリックし、画像のソート方向アイコン imgSortDir に従って、2 つのソート関数のいずれかを呼び出すローコードアクションスクリプトを追加します。



Edit をクリックすると Action Editor ポップアップウィンドウが表示されます。左側のパレットに利用可能なローコードアクションノードがあることに注意してください。パレットからノードをドラッグ&ドロップして並べ替えるロジックを構築します。



- CONDITON の下の If ノードを Start ノードの下のエディタにドラッグします。

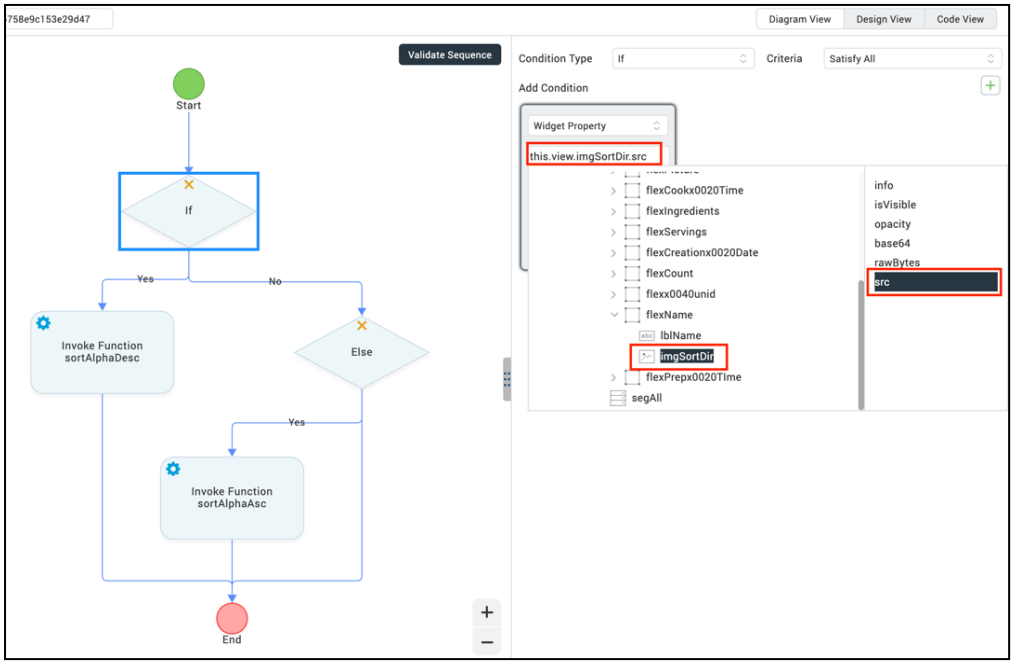
- If ノードは imgSortDir ウィジェットの画像を調べて、どのソート関数を呼び出すかを決定します。If 条件に次のプロパティを設定します。

    a) Condition Type: If

    b) Criteria: Satisfy All

    c) Widget Property を選択します。

    d) ポップアップナビゲーションツールを使用して this.view.imgSortDir.src を選択します。



    e) Equals を選択します。

    f) Value を選択します。

    g) imgchevronup.png を入力します。

- Invoke Function ノードを Yes ブランチとしてエディタの If ノードの左下にドラッグします。

- 関数 sortAlphaDesc を選択します。理由は、If ノードで上矢印アイコン imgchevronup.png（Z から a への降順ソート用）をテストしているためです。

- • Else ノードを No 分岐として If ノードの右下にドラッグします。パスが上記のスナップショットと一致しない場合はノードを右クリックして削除できます。
- Invoke Function ノードを Else ノードの下にドラッグします。
- 関数 sortAlphaAsc（a から Z への昇順ソート用の下矢印）を選択します。
- ローコードアクションスクリプトが上のスナップショットと一致していることを確認し、Save をクリックします。
- Iris プロジェクトを保存します。

4. これでプロジェクトのプレビュービルドを実行する準備ができました。Iris のトップレベルメニューから Build -> Live Preview Settings をクリックします。
5. Live Preview Settings で Responsive Web チャンネルのみがチェックされていることを確認します。Clean Preview にチェックを入れます。ビルドモードは Debug です。Foundry 環境が正しいことを確認し、必要であれば Change をクリックして設定します。最後に、Save & Run をクリックして Web アプリをビルドして実行します。
6. プレビュービルドが完了すると、Volt MX Go Iris Preview デバッグブラウザのポップアップが表示されます。そこでログインしてレシピアプリをテストできます。デバッグブラウザは閉じてかまいません。テストでは別の任意のブラウザで行います。
7. 任意のブラウザからプレビューアプリをテストするには次のようにします。
   - Volt Iris IDE の中央下部に、Console、Search、Build、Debug、Test、Terminal のタブがあります。Console をクリックします。
   - メッセージウィンドウを一番下までスクロールします。次のリンクがあるはずです。Response web: http://127.0.0.1:9989/<あなたの Iris プロジェクト名>/kdw

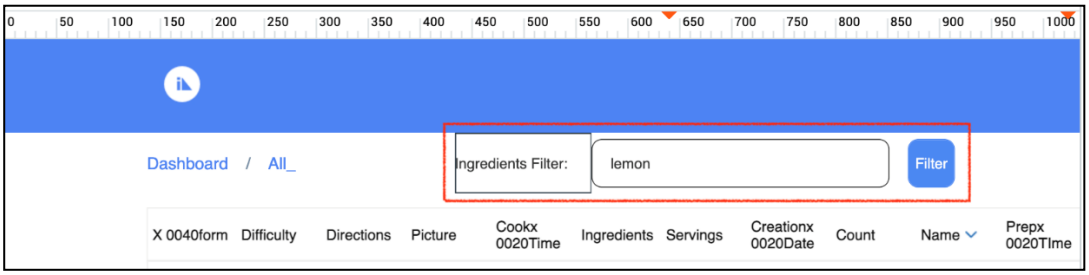
レシピアプリの新しいソート機能をテストしてみましょう。レシピのリストが適切に並べ替えられましたか？

レシピのフィルタリングの実装

viewAllGrid というフォームにレシピのフィルタリングを実装します。レシピをフィルタリングするには、ユーザーはフィルタリングテキストボックスにキーワード（例：レモン）を入力し、フィルタをクリックします。キーワードを含むすべてのレシピの原材料を検索し、原材料にキーワードを含むレシピの新しいリストを作成し、新しいレシピリストでフォーム viewAllGrid を更新します。

ステップ

1. Volt Go Iris IDE から、IDE が、前のレッスンで Design Import によって作成された Recipe プロジェクトですでに開かれていることを確認します。Iris キャンバスの Design ビューにいることを確認します。
2. IDE の左上側で Project タブの下のフォームツリー構造に移動して、フォーム viewAllGrid を開きます。フォームが Iris のセンターキャンバスに表示されているはずです。
3. フィルターキーワードを受け付けるテキストボックスと、材料をフィルターするボタンを追加します。すべてのレシピ材料を検索し、新しいレシピリストを作成します。そしてセグメントウィジェットの segAll を更新して、新しいリストからレシピをリストアップします。



- プロジェクトツリーパス Project -> Responsive Web / Desktop -> Forms -> viewAllGrid にある FlexContainer の flexBreadcrumb をクリックして Iris にフォーカスを設定します。
- Default Library から FlexContainer ウィジェットを中央のキャンバスにドラッグします。
- 右クリックして、新しい FlexContainer の名前を flxFilterContainer に変更します。
- PROPERTIES -> Look に移動して、次の Look プロパティを設定する：

    a) Left: 16%,

b) Top: 0 Dp,

c) Width: 60%,

d) Height: 100%,

- PROPERTIES -> FlexContainer -> Layout Type に移動します。
- flxFilterContainer の Layout Type を Flow Horizontal に設定します。
- ラベルウィジェットを Default Library から中央のキャンバスにドラッグします。
- Navigate to PROPERTIES -> Look -> Text に移動します。
- 新しいラベルウィジェットのテキストに“Ingredients Filter:”と入力します。
- Iris のフォーカスを flxFilterContainer に設定します。
- TextBox ウィジェットを中央のキャンバスにドラッグします。これによりフィルターのキーワードを受け付けられるようになります。
- Right-click the new TextBox and rename it to txtFilter,
- 新しいテキストボックスを右クリックして、名前を txtFilter に変更します。
- Navigate to PROPERTIES -> Look to set the following Look properties:
- PROPERTIES」→「Look」に移動して、以下の Look プロパティを設定します：

a) ユーザーのためのキーワード例として、テキストを lemon に設定します。

b) Left: 0 Dp,

c) Top: 0 Dp,

d) Width: 50%,

e) Height: 80%,

f) Center Y: 50%.

- PROPERTIES -> Skin を開き、txtFilter に Rounded Corner スタイルでボーダーを追加します。

a) Border -> Size: 1 Px

b) Border -> Style: Rounded Corner

c) Fonts -> Size: 88%

- Iris のフォーカスを flxFilterContainer に設定します。
- Button ウィジェットを中央のキャンバスにドラッグします。これにより内容のフィルタリングが開始されます。
- 新しい Button で右クリックし、名前を btnFilter に変更します。
- PROPERTIES → Look に移動し、以下の Look プロパティを設定します。

a) Text: Filter

b) Left: 3%

c) Top: 0 Dp

d) Width: 8%

e) Height: 80%

f) Center Y: 50%.

- PROPERTIES -> Skin に移動して、txtFilter にスタイル Rounded Corner を使用してボーダーを追加します。

a) Border -> Size: 1 Px,

b) Border -> Color: #4b88f1,

c) Border -> Style: Rounded Corner,

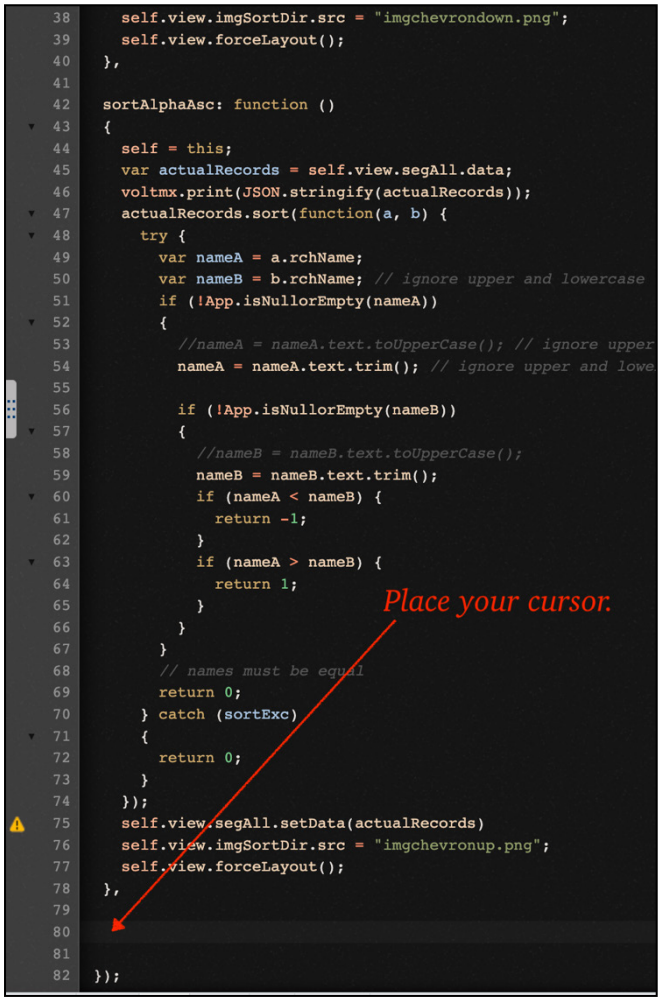
d) Fonts -> Size: 88%,

- Iris プロジェクトを保存します。

4. フォーム viewAllGrid のコントローラモジュール viewAllGridController に JavaScript 関数をひとつ追加します。この関数の名前は filterData で、既存のレコードを ingredient キーワードでフィルタリングし、更新されたレシピリストを返します。フィルタリングされる前

のレシピリストを保存するためにグローバル変数も追加します。新しいキーワードフィルタがリクエストされたときに保存されたリストを使用します。

- Project -> Responsive Web / Desktop -> Controllers -> viewAllGridController に移動します。
- viewAllGridController をクリックして中央のキャンバスで開きます。コントローラには上記の 2 つの並べ替え関数が含まれています。カーソルをモジュールの一番下、2 番目の並べ替え関数の下、最後の行の上に移動します。



- 下のテキストボックスから関数 filterData をコピーし、viewAllGridController エディタの 2 番目のソート関数の下に貼り付けます。

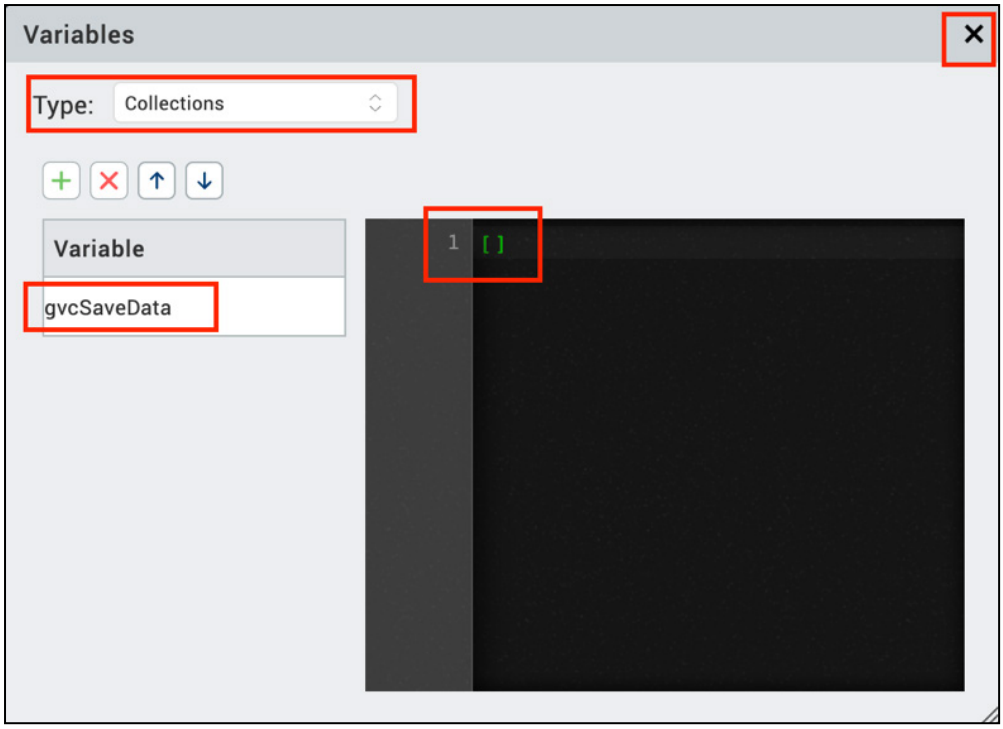
```
filterData: function (inKeyword)
{
  self = this;
  var actualRecords = self.view.segAll.data;
  if (gvcSaveData.length > actualRecords.length)
  {
    actualRecords = gvcSaveData;
  }
  var arFilteredRecords = [];
  for (index=0; index < actualRecords.length; index++)
  {
    let tmpIngredient = actualRecords[index].rchIngredients.text;
    if (tmpIngredient.indexOf(inKeyword) > -1)
    {
      arFilteredRecords.push(actualRecords[index]);
    }
  }

  if (gvcSaveData.length < 1)
  {
    gvcSaveData = actualRecords;
  }
  self.view.segAll.setData(arFilteredRecords)
  self.view.forceLayout();
},
```

- Iris プロジェクトを保存します。

注意: JS 関数 filterData では、self.view.segAll はセグメント ウィジェットであり、Iris プロジェクトで既存のすべてのレシピを viewAllGrid 形式でリストするために使用されます。グローバル変数 gvcSaveData はフィルタリングが適用される前のオリジナルのレシピリストを保存します。次に、このグローバル変数 gvcSaveData を作成します。

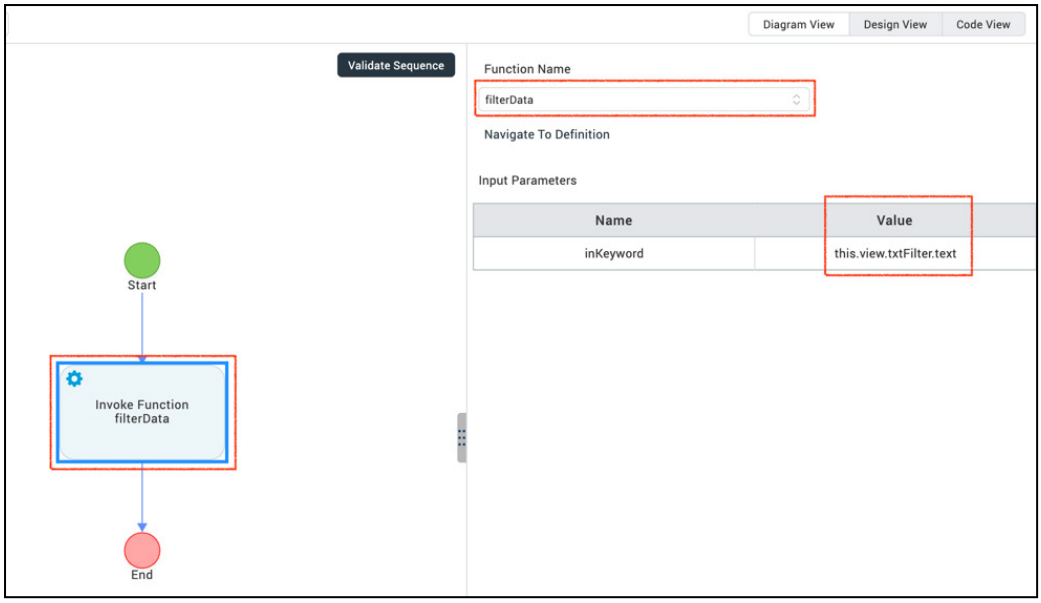
5. グローバル変数 gvcSaveData を追加します。

- At the top of Iris, click the top level menu-item Edit -> Global Variables,
- Iris の一番上のメニュー、Edit -> Global Variables をクリックします。
- Type を Simple から Collections に変更します。
- Variable というラベルの下に gvcSaveData と入力します。入力ボックスが表示されない場合は + 記号をクリックします。
- 隣接する 1 行のテキストボックスに下図のように "[]" と入力します。

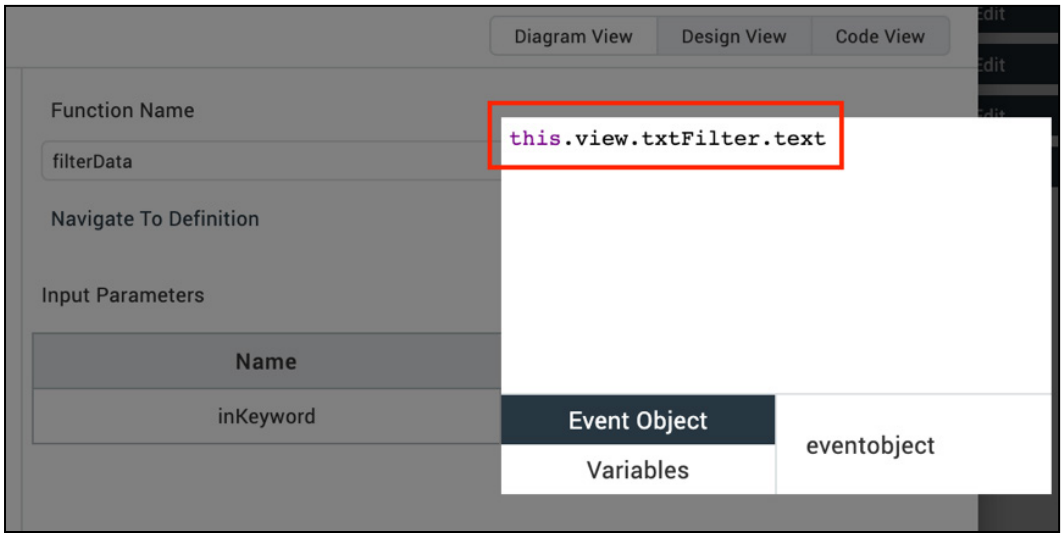


- Iris プロジェクトを保存します。

6. ユーザーによる Filter ボタンのクリックを関数 filterData に接続します。そのために、このボタンの onClick イベントにローコードアクションスクリプトを追加します。
7. このボタンの onClick イベントにローコードアクションスクリプトを追加します。
    - Iris のフォーカスをボタン btnFilter に設定します。
    - PROPERTIES -> Action -> onClick -> Edit に移動します。
    - Action Editor で Invoke Function を左側のパレットから Start ノードの下にドラッグします。
    - Function Name で関数リストから関数 filterData を選択します。
    - Input Parameters で Value ラベルの下の Edit をクリックします。



- 入力パラメータの値として this.view.txtFilter.text を入力します。filterData 関数が呼び出されると、入力テキストボックス txtFilter のキーワードがフィルタリングに使用されます。



- ポップアップウィンドウの外側をクリックしてポップアップ ウィンドウを閉じます。
- Save をクリックしてアクションスクリプトを保存しポップアップを閉じます。
- Iris プロジェクトを保存します。

8. これで Iris プロジェクトのプレビュービルドを実行し、新しいフィルター機能をテストする準備ができました。ステップ 4 から 7 を参照してアプリをビルドし、ブラウザでテストしてください。

レシピをフィルターできますか？

まとめ

このレッスンを完了しました。この時点で、あなたのデザインインポートレシピアプリは 2 つの新機能を持っています。すべてのレシピのリストで、Domino の All ビューを経由して、レシピ名を昇順または降順に並べ替えることができます。特定の食材でレシピリストをフィルターし、ユーザーが指定したフィルターの食材を含むレシピのみを表示することができます。

法的ステートメント

このエディションは、HCL Volt MX Go のリリース 2.0.1、および新しいエディションで別段の記載がない限り、それ以降のすべてのリリースおよび変更に適用されます。

あなたが HCL Technologies Ltd.に情報を送信する場合、あなたは HCL Technologies Ltd.に、あなたに対していかなる義務を負うことなく、適切と思われる方法で情報を使用または配布する非独占的な権利を付与します。

米国政府ユーザーへの注意 - 制限された権利に関連する文書 - 使用、複製、または開示は、HCL Technologies Ltd.との GSA ADP スケジュール契約に規定された制限に従うものとします。

免責事項

本レポートは、HCL 利用規約（https://www.hcl.com/terms-of-use）および以下の免責事項の対象となります：

本レポートに含まれる情報は、情報提供のみを目的としています。本レポートに含まれる情報は、情報提供のみを目的として提供されるものであり、本書に含まれる情報の完全性および正確性を確認するよう努めたが、商品性、非侵害性、特定目的への適合性の黙示保証を含むがこれに限定されない、明示または黙示を問わずいかなる保証もなく、現状のまま提供されるものである。また、本情報は、HCL 社の現在の製品計画および戦略に基づいており、HCL 社により予告なく変更される場合があります。HCL は、本レポートまたはその他の資料の使用またはその他の関連から生じる直接的、間接的、偶発的、結果的、特別またはその他の損害について責任を負わないものとします。本書に含まれるいかなる内容も、HCL 社またはその供給業者やライセンサーによる保証や表明を意図するものではなく、またそのような効果をもたらすものでもありません。

本レポートにおける HCL の製品、プログラム、サービスへの言及は、HCL が事業を展開するすべての国でそれらが利用可能になることを意味するものではありません。本プレゼンテーションで言及されている製品のリリース日や機能は、市場機会やその他の要因に基づき、HCL の独自の裁量で随時変更される可能性があり、将来の製品や機能の提供を約束するものではありません。これらのレポートをサポートするために使用される基礎データベースは、毎週更新されます。この Web ツールを使用して生成されたレポートと他の HCL ドキュメンテーションソースの間に見られる不一致は、このツールと他のソースの公開および更新サイクルが異なることに起因する場合も、そうでない場合もあります。本レポートに含まれるいかなる内容も、あなたが行った活動が特定の売上、収益の増加、節約、またはその他の結果をもたらすことを意図したものではなく、またそのような効果を持つものでもありません。利用者は、本レポートの結果として利用者が得た結果または利用者が行った決定について、単独で責任を負うものとします。HCL 利用規約(https://www.hcl.com/terms-of-use)にかかわらず、本サイトの利用者は、本ツールから生成されたレポートを利用者自身の内部業務目的のためにコピーおよび保存することが許可されています。それ以外の使用は許可されません。