

Apogee Form Components

v 1.0.0-p2

In Apogee there is a configurable form widget that is used in several places to easily define forms for the user. This document gives the reference information for using these configurable forms.

Components Using the Configurable Form

Two examples where the form widget is used are the components *Action Form Cell* and *Data Form Cell*.

A *Form Action Cell* is like a non-modal dialog box. Here the programmer configures the layout of the form, including a submit button to take action. The programmer defines a handler for the save button which defines the action. There is no saved data associated with the *Form Action Cell*.

A *Form Data Cell* is similar to a *Data Cell* except the user enters data into a form rather than text editor. The configurable form format is defined by the programmer. When the user changes the value, the *save bar* appears just as if the user changed the value of a data table. When the user saves, the data is saved to an internal data table which can be accessed from the code.

In either of these components, the programmer must define the layout for the form, which is described in this document.

Configurable Panel and Configurable Elements

The name of the configurable form is actually *ConfigurablePanel*. It includes a list of entries, which are called *ConfigurableElements*. There are several types of configurable elements.

Here we will demonstrate a single form with several elements on it. This is from a *Form Action Cell*.

The following code is a function body that returns the layout for a form. The layout is basically a JSON but it can include some functions. This may be submit functions, if it has them, or it could be other handlers.

```
let onSubmit = formValue => admin.getCommandMessenger().dataCommand("FormOutput",formValue);
let onCancel = () => alert("Never mind");
```

```

return [
  {
    type: "heading",
    level: 2, //1-6
    text: "Elements in Alphabetical Order (except this one, and the last one)"
  },
  {
    type: "checkbox",
    label: "Whip it: ", //optional
    value: true, //optional
    key: "whipit"
  },
  {
    type: "checkboxGroup",
    label: "When to Whip It: ", //optional
    entries: [
      ["When a problem comes along", "problem"],
      ["Before the cream sits out too long", "cream"],
      ["When something's going wrong", "problem"]
    ], //here we express as display value + return value
    //horizontal: true, //optional, defaults to false
    key: "responso"
  },
  {
    type: "dropdown",
    label: "Actions: ", //optional
    entries: [
      "Shape it up", "Get straight", "Go forward", "Move ahead", "Try to detect it"
    ], //see checkboxGroup for alternate choice format
    value: "Shape it up", //optional
    key: "choices"
  },
  {
    type: "htmlDisplay",
    html: "<em>When a good time turns around you must whip it. You will never live it down unless you whip it. No one gets away until they whip it</em>"
  },
  {
    type: "invisible",
    value: {
      "msg": "Whip it carefully",
      "deliveryTone": "silent"
    },
    key: "secret"
  },
  {
    type: "radioButtonGroup",
    label: "More, apparently random, actions: ",
    entries: [
      "Crack that whip", "Give the past a slip", "Step on a crack", "Break your momma's back"
    ], //We can also use [
      ["Yes", "yes"], ["No", "no"]
    ] which gives the display value and the return value
    value: "cream"
  },
  {
    type: "spacer",
    height: 20 //optional, defaults to 15, in pixels
  }
]


```

```

    },
    {
      type: "panel",
      formData: [ //a panel includes a list of elements, like the form itself
        {
          type: "textField",
          //password: true, //optional - if this is set to true it is a password field
          label: "Title: ",
          value: "Whip It",
          size: 50, //optional character width of field
          key: "title"
        },
        {
          type: "textarea",
          label: "Lyrics: ",
          value: "Crack that whip\nGive the past a slip\nStep on a crack\nBreak your momma's
back\n\nWhen a problem comes along\nYou must whip it\nBefore the cream sets out too long\nYou must whip
it\nWhen something's going wrong\nYou must whip it\nNow whip it\nInto shape\nShape it up\nGet straight\nGo
forward\nMove ahead\nTry to detect it\nIt's not too late\nTo whip it\nWhip it good\n\nWhen a good time
turns around\nYou must whip it\nYou will never live it down\nUnless you whip it\nNo one gets away\nUntil
they whip it\n\nI say whip it\nWhip it good\nI say whip it\nWhip it good\n\nCrack that whip\nGive the past
a slip\nStep on a crack\nBreak your momma's back\n\nWhen a problem comes along\nYou must whip it\nBefore
the cream sets out too long\nYou must whip it\nWhen something's going wrong\nYou must whip it\n\nNow whip
it\nInto shape\nShape it up\nGet straight\nGo forward\nMove ahead\nTry to detect it\nIt's not too late\nTo
whip it\nInto shape\nShape it up\nGet straight\nGo forward\nMove ahead\nTry to detect it\nIt's not too
late\nTo whip it\nWell, whip it good",
          rows: 10, //optional number of rows
          cols: 50, //optional character width of field
          key: "lyrics"
        }
      ],
      key: "lyricsPanel",
      selector: { //any item can include a selector, which hides or shows it based on a parent element
and its value
        parentKey: "whipit",
        parentValue: true,
        //keepActiveOnHide: true //if this is missing or false the value of a hidden panel is not
included in the return value
      }
    },
    {
      type: "submit",
      onSubmit: onSubmit, //optional - if this is present the button is present
      onCancel: onCancel, //optional - if this is present the button is present
      submitLabel: "Whip it!", //optional, defaults to OK if button is present
      cancelLabel: "Never mind", //optional, defaults to Cancel if button is present
      //submitDisabled: true,
      cancelDisbled: true,
      key: "xxx" //this isn't really needed
    }
  ]
}

```

Here is the output of this layout:

 **exampleForm** [vForm](#) [>Code](#) [>Private](#)

Form

Elements in Alphabetical Order (except this one, and the last one)

Whip it: ☒

When to Whip It:

☐ When a problem comes along

☐ Before the cream sits out too long

☐ When something's going wrong

Actions:

When a good time turns around you must whip it. You will never live it down unless you whip it. No one gets away until they whip it

More, apparently random, actions:

☐ Crack that whip

☐ Give the past a slip

☐ Step on a crack

☐ Break your momma's back

Title:

Lyrics:

Crack that whip
Give the past a slip
Step on a crack
Break your momma's back

When a problem comes along
You must whip it
Before the cream sets out too long
You must whip it
When something's going wrong

Element Types

The section gives the available types of elements and the configuration rules for each. There are also some additional configuration options listed below in the section [Shared Configuration Options](#).

CheckboxElement

This is a simple, singular check box.

Init Data:

- type - "checkbox"
- label - The label for the checkbox (optional)
- value - initial value: true/false (optional)
- key - The key used to report this value in the form result
- onChange(elementValue,formObject) - handler called when the value of this checkbox changes (optional)

CheckboxGroupElement

This is a group of checkboxes.

Init Data:

- type - "checkboxGroup"
- label - The label for the checkbox group (optional)
- entries - array of values, one for each checkbox (optional)
- value - a list containing the values for the checked checkboxes
- key - The key used to report this value in the form result
- horizontal - if this is set to true the checkboxes will be arranged horizontally, to the extent this is possible. Otherwise they will be placed one to each line. (optional)
- onChange(elementValue,formObject) - handler called when the value of this checkbox changes (optional)

DropdownElement

This is a dropdown element.

Init Data:

- type - "dropdown"
- label - The label for the dropdown (optional)
- entries - array of values in the dropdown
- value - this initial value for the dropdown (optional)
- key - The key used to report this value in the form result

- onChange(elementValue,formObject) - handler called when the value of this checkbox changes (optional)

HtmlDisplayElement

This is an element that displays arbitrary HTML.

Init Data:

- type - "htmlDisplay"
- html - This is the HTML string

HeadingElement

This is a heading element.

Init Data:

- type - "heading"
- text - This is the text for the heading
- level - This gives the weight of the heading, like the weight in an HTML heading.

InvisibleElement

This is an element that holds a value but is non-interactive and not visible.

Init Data:

- type - "invisible"
- value - The value for the element
- key - The key for the element

PanelElement

This is an panel that contains a list of child elements.

InitData

- type - "panel"
- formData - This is the init data for the form, in the same format as the init data for the parent panel.
- onChange - This is a handler for when the element changes. It differs from the onChange handler for the form in the panel only in that the panel form onChange function includes the panel entry as an argument. The Element on onChange function includes the parent panel as an argument, just as is done on the other elements.
- key - The key for the element

RadioGroupElement

This is a group of radio buttons.

Init Data:

- type - "radioButtonGroup"
- groupName - the HTML radio button group name
- label - The label for the radio button group (optional)
- entries - array of values, one for each radio button
- value - the value of the initially checked button (optional)
- key - The key used to report this value in the form result
- horizontal - if this is set to true the checkboxes will be arranged horizontally, to the extent this is possible. Otherwise they will be placed one to each line. (optional)
- onChange(elementValue,formObject) - handler called when the value of this checkbox changes (optional)

SpacerElement

This is an element to add some vertical space to the form.

Init Data:

- type - "spacer"
- height - pixel height for the space (optional - current default = 15px)

TextFieldElement

This is a text field.

Init Data:

- type - "textField"
- label - The label for the text field (optional)
- value - the initial value (optional)
- key - The key used to report this value in the form result
- password - if this is true, the field is a password field (optional)
- size - the character width of the text field. (optional)
- onChange(elementValue,formObject) - handler called when the value of this element changes (optional)
- onChangeCompleted(elementValue,formObject) - handler called when the element loses focus and was changed (optional)

TextareaElement

This is a text area.

Init Data:

- type - "textarea"
- label - The label for the text field (optional)
- value - the initial value (optional)

- key - The key used to report this value in the form result
- rows - The number of rows in the text area. (optional)
- cols - The number of columns in the text area. (optional)
- onChange(elementValue,formObject) - handler called when the value of this element changes (optional)
- onChangeCompleted(elementValue,formObject) - handler called when the element loses focus and was changed (optional)

Custom Element

This is a custom element. It is made by explicitly constructing a configurable element so it has the proper functionality. (This is done because at the time there is no support for "class" in the code. If this is added you can pass in a constructor that extends ConfigurableElement).

Init Data:

- type - "custom"
- key - The key used to report this value in the form result
- builderFunction - This is a function which converts a base ConfigurableElement into the custom element.

SubmitElement

This element provides a submit button and a cancel button, to control the panel

Init Data:

- type - "submit"
- key - The key used to identify the element
- submitLabel - a label for the submit button. Default is "OK"
- cancelLabel - a label for the second button. Default is "Cancel"
- submitDisabled - this enables or disables the submit button (optional)
- cancelDisabled - this enables or disables the cancel button (optional)
- onSubmit(formValue,formObject) - handler called when the value of this element changes (optional)
- onCancel(formObject) - handler called when the element loses focus and was changed (optional)

Additional Element Functions

- submitDisable(boolean) - this enables or disables the submit button
- cancelDisable(boolean) - this enables or disables the cancel button

Shared Configuration Options

In addition to the above listed initialization options, there are some additional options common to all elements.

- **state** - that state for the form element.
 - **STATE_NORMAL** - visible and not disabled
 - **STATE_DISABLED** - visible but disabled
 - **STATE_HIDDEN** - not visible.
 - **STATE_INACTIVE** - not visible. In this state the element value is NOT included in the form result.
- **selector** - This can be used to display an element based on a parent selection. It is most often used for panels but can be used for any element. This is an object with the following fields.
 - **parentKey** - The key for the parent. The parent must be in the same panel and be before the child.
 - **parentValue** - This is the value of the parent that triggers display of the child.
 - **keepActiveOnHide** - If this is true the form value will return the value of any element whether it is hidden or not. This is how a tab would typically work. If the value is false or undefined then only the visible panel is included in the form value.
- **onChange(elementValue,formObject)** - A standard event for elements. It is further described under each element.

Additional Reference

In more advanced usage of forms, such as if you are incorporating one in a custom form.

Configurable Form Panel

Methods

- **constructor(optionalContainerClassName)** - Creates the form, passing the initialization value. Optionally, a container CSS class can be specified.
- **configureForm(formInitData)** - This method constructs the form inside the panel. It can be called as many times as desired to reconfigure the panel.
 - **formInitData format** - This is a javascript/json object with the following fields:
 - **layout** - (REQUIRED) This is an array of initialization elements defining the elements in the form.
 - **onSubmit** - (OPTIONAL) This is a javascript/json object that adds a submit button to the form. The submit button may also be added as an individual layout element. The fields in this object are:

- onSubmit - This function is a handler for the submit action. If it is not included there is no submit button.
- onCancel - This function is a handler for the cancel action. if it is not included there is no cancel button.
- submitLabel - This is an optional label for the submit button.
- cancelLabel - This is an optional label for the cancel button.
- onChange - (OPTIONAL) This handler is called whenever the form changes. The arguments are (formValue,formObject).
- disabled - (OPTIONAL) If this field is set to true, the form will be disabled.
- getEntry(key) - This gets a Form Element, by key, from the panel.
- getValue() - This gets the value for the form.
- setValue(formValue) - This sets the value for the form.
- getElement() - This returns the DOM element for the panel.
- getChildEntries() - This returns an array of form elements.
- addSubmit(onSubmit,onCancel,optionalSubmitLabel,optionalCancelLabel) - This is an additional way of adding a submit entry to a form, besides using the initData for the panel.
- addOnChange(onChange) - This is called if the form changes value. The onChange function is passed the arguments (formValue, formObject)
- setDisabled(isDisabled) - This enables or disables the form.

Configurable Form Elements

Methods

The following functions are available on each form element:

- getKey() - returns the key for this element
- getState() - returns the state for this element
 - the form value. (In all other states the value is included)
- setState(state) - sets the state for this element
- setValue(value) - sets the value for this element
- getElement() - returns the DOM element for this for element
- getForm() - returns the parent form for this element

Components

Action Form Cell

This component displays a form which is dynamically configured by the user. It is recommended it is used to take an action, as opposed to just saving data. If saving data is desired, the Action Form Cell is a good choice.

This component includes the following views:

- Form - this displays the form
- Code - This should be used to enter a function body returning the form initialization info. along with a private code display. For the format, see the definition of the argument "formInitData" of the "configureForm" function in the ConfigurableForm documentation.
- Private - This allows for an private constants or functions needed by the function body initializing this form.

Data Form Cell

This component also includes a configurable form. This one however behaves something like a Data Cell where the end user edits on the form rather than a text editor. No submit element is needed. A "save" button, like on a text editor, will appear. When save is pressed the data is sent to an internal data cell.

The Data Form Cell actually maps to a folder object, to which the user can not add tables. The folder contains three internal tables:

- layout - This method should return the layout info for the form. For the format, see the definition of the argument "formInitData" of the "configureForm" function in the ConfigurableForm documentation.
- data - This table holds the form value. It can be used both to initialize the form and read the form result.
- isValid - This is a function that is called with the argument formValue. When the form is submitted this is called to validate the data. It should return one of the following values. If a different value is returned, an error message is shown.
 - true (boolean value) - This means the data is valid
 - error message (string) - This error message will be shown to the user and the form data will not be saved.

This component includes the following views:

- Form - This displays the form.
- Layout Code - This should be used to enter a function body returning the form initialization info. along with a private code display. For the format, see the definition of the argument "formInitData" of the "configureForm" function in the ConfigurableForm documentation.
- Layout Private - This allows for an private constants or functions needed by the function body initializing this form.
- isValid(formValue) - This view allows for entry of the isValid function
- isValid Private - This allows for an private constants for functions needed by the isValid function.

- Form Value - This shows the form value. It can be edited to set the form value, or used to read the return value of the form.