BUCKET INPUT DIRECTIVE

FORM FACTORY V2

The following document will explain the usage of the input bucket directive.

Input bucket directive attributes:

- **bucketMode** - (=)
  - *boolean*, Inputs can only be selected once
- **multiStepMode** - (=)
  - *boolean*, Inputs grouped by step number
- **displayTitle** - (=)
  - *boolean*, Whether to display the input directives default title. Can be overridden by setting the following resource bundle key: **angular.ffInputBucketDirective.label.selectInput**
- **setDefault** - (=)
  - *boolean*, If true, the first available input will be selected if no previous value was selected.
- **fieldName** - (=)
  - *string*, represents the previously selected input field name. Also used to initialize **selectedInput**.
- **emitOnChange** - (@?)
  - *string*, The name of a function that should be emitted once an input has been selected. Will contain the **selectedInput** as the *2nd parameter*
- **broadcast** - (&?)
  - A function that will be called once an input has been selected. When the provided function is called you can, for example perform a broadcast to alert any listeners about the change.
- **availableInputs** - (=)
  - *array,* A prepared array containing objects that describe the represented inputs. Required properties are:
    - **label** - The label identifying the input, that is display to the user.
    - **fieldName** - The input name.
    - **index** - Index of the input object in the **availableInputs** array. Used to track the selected inputs.
    - **stepIndex** - Used to sort inputs when using **multiStepMode**.
    - **stepName** - Used as the group name when grouping inputs.

Example of initializing the input bucket directive's **availableInputs** property:

```
scope.availableInputs = [];
_.each(scope.form.steps, function(step, sIndex) {
  _.each(step.inputs, function(input) {
     scope.availableInputs.push({
        fieldName: input.name,
        label: input.label,
        stepName: step.label,
        stepIndex: sIndex
     });
     scope.availableInputs[scope.availableInputs.length-1].index = scope.availableInputs.length-1;
  });
});
scope.selectedInput = scope.availableInputs[0];
```

In the above example, we would be preparing the **availableInputs** to be used in a **multiStepMode**, as we are supplying the **stepName** and **stepIndex**.

In the template we would call the directive but supplying it with the correct attributes as follows:

```
<ff-input-bucket display-title="true"
         field-name="selectedInput.fieldName"
         bucket-mode="false"
         multi-step-mode="true"
         emit-on-change="notifyInputSelected"
         available-inputs="availableInputs">
</ff-input-bucket>
```

By doing so, we specify to the directive that it will not function as a bucket, It will display the inputs grouped by their corresponding step. Upon input selection the input bucket directive will emit a call corresponding to **notifyInputSelected**. In the directive that uses the input bucket directive we can then listen for call that was emitted:

```
//Emitted from Input bucket, the function will contain
//the selectedInput as the 2nd parameter.
scope.$on('notifyInputSelected', function(event, selectedInput) {
  //Do something here
});
```

When an input is selected, it is attached to the current scope as **selectedInput**, that will contain the object that is selected from the **availableInputs** array. As mentioned in the comments of the above code snippet, the **selectedInput** is also available as the 2nd parameter of the emitted function.

This is a simple scenario to showcase the usage of the input bucket directive. More complex scenarios can be achieved by using the input bucket directive in bucket mode.