Form Component

What is a Form Component (form)

Form Component in the Solution Explorer Tree is a special contained form used for simplified design and runtime optimization for the NGClient. The same behavior can be achieved via tabless tabpanel, but form component has optimized display because all elements are part of the same form, taking away the overhead of tabpanel and loading of another form.

Such a Form Component Form can be places on the form by UI Components which do have a model property that is of type 'formcomponent', Servoy ships for absolute and the bootstrap layout a FormComponentContainer component. This is a UI Component with a single 'formcomponent' property that displays the form as a a tabless tabpanel.

A Form Component Form can't have its own logic (js file), it is also not available at runtime under "forms.xxx" because there is no instance of such a form, Only the template and its properties are used, events like onclick of a button can be assigned but only to scopes or entity methods (if the datasource is specified). Also because it does not have its own instance at runtime also all the form component events and commands like onLoad are not available for it.

Runtime usage

On a main form that has added a FormComponentContainer component and assigned the property to a form component form can change all the elements properties of the form it includes that has a name. So naming the element on a Form Component form will create the public API of that form.

As an example of the FormComponentContainer component that has a model property: "containedForm" : "formcomponent", that you placed on the form with the name "fc1" that has form assigned that has 2 buttons called "btn1" and "btn2". At runtime you can access these as:

elements.fc1.containedForm.btn1.text = "Hallo"

In the designer you can do the same thing by selecting the "btn1" in the designer or through the outline view and then changing the "text" property to a certain value. You only change the value of that form component form for that specific instance of 'fc1' on the main form. So you can place another FormComponentContainer component call that "fc2" point to the same Form Component form and then alter for that instance the "btn1" to something else:

elements.fc2.containedForm.btn1.text = "Hallo2"

So this way you can show 1 form (form component) twice at the same time showing different stuff.

Solution model

for the main form is almost the same:

```
var mainForm = solutionModel.getForm("mainform");
var fc = mainForm.getWebComponent("fc1");
fc.setJSONProperty("containedForm", solutionModel.getForm("myformcomponentform"));
fc.setJSONProperty("containedForm.btn1.text", "Hallo");
```

What can't be done is changing the form component form through solution model, you can get it through solutionModel.getForm("myformcomponentform"), you can read all kinds of properties from itself or its components, but you are not allowed to alter it so adding or changing components on it. This is because the template and its state is, or can be, shared across multiply forms on many places. So altering such a thing would mean that recreateUI has to be called on all of them, and we have to push new templates of those changes also to all of them. This would result in way less efficient caching and reuse of stuff.

Creating a FormComponentContainer component

The most simple one, that is like a tabpanel only has one property:

The spec file of a component using a Form Component Form

```
{
	"name": "servoycore-formcomponent",
	"displayName": "FormComponentContainer",
	"version": 1,
	"icon": "servoycore/formcomponent/form.gif",
	"definition": "servoycore/formcomponent/formcomponent.js",
	"libraries": [],
	"model":
	{
		"containedForm": "formcomponent"
	}
}
```

Then the js file watches that property and calls the servoyApi.getFormComponentElements() and appends te result directly to its element

```
angular.module('servoycoreFormcomponent',['servoy']).directive('servoycoreFormcomponent', [function() {
    return {
           restrict : 'E',
           scope : {
                   model: '=svyModel',
                   api : "=svyApi",
                   svyServoyapi: "=",
                   handlers: "=svyHandlers"
           },
           controller: function($scope, $element, $attrs)
           {
                   $scope.$watch("model.containedForm", function(newValue) {
                           $element.empty();
                           if (newValue) {
                                   var elements = $scope.svyServoyapi.getFormComponentElements("containedForm",
newValue);
                                           $element.append(elements);
                           }
                           else {
                                   $element.html("<div>FormComponent, select a form</div>");
                           }
                   });
        }
    }
}])
```

A splitpane could have 2 properties instead of that containedForm property shown above like "left" and "right" that then shows left and right a form.

Extra properties

The form component gets from the system on its formcomponent property value a few extra properties that it can use to show the elements.:

formHeight: the height of the contained form.

formWidht: the height of the contained form.

absoluteLayout: Is the contained form an absolute layout form (false means it is a reponsive form)

Below is the example how the bootstrap (reponsive) version of a form component implemenation works:

So if a height or widht is given then it will generate everything inside a div, with that specific height. If no height is given but the containedForm is in absoluteLayout it will use the form height and width to generate that wrapping div.

```
function createContent() {
        $element.empty();
        var newValue = $scope.model.containedForm;
        if (newValue) {
                var elements = $scope.svyServoyapi.getFormComponentElements("containedForm", newValue);
                var height = $scope.model.height;
                var width = $scope.model.width;
                // if the form of the form component is a absolute layout
                // make sure to get the height and width from that form if our own height/widht are not set.
        // so that an absolute form is always inside a div that is generated below
                if (newValue.absoluteLayout) {
                        if (!height) height = newValue.formHeight;
                        if (!width) width = newValue.formWidth;
                }
                if (height || width) {
                        var template = "<div style='position:relative;";
                        if (height) template += "height:" +height + "px;"
                        if (width) template += "width:" +width + "px;"
                        template += "'";
                        if ($scope.model.styleClass)  template += " class='" +$scope.model.styleClass + "'";
                        template += "></div>";
                        var div = $compile(template)($scope);
                        div.append(elements);
                        $element.append(div);
                }
                else $element.append(elements);
        }
        else {
                $element.html("<div>FormComponentContainer, select a form</div>");
        }
}
```