jahia

USING CALLBACKS

FORM FACTORY V2

WARNING: If you copy-paste code from this document you will have to make sure that all characters are properly transferred. Pay close attention to hyphens. Also, if you create your views using Studio make sure you remove all taglibs that get generated by default.

IMPORTANT: When working with JavaScript it may be necessary to flush browser cache after changes to Javascript files. Most modern browsers provide an add-on tool to perform cache flushes. To get such a tool for your browser you should search available add-ons with a search criteria such as "clear cache" or "empty cache", select appropriate add-on and follow given instructions.

Global callbacks

Global callbacks allow you to process some action after form has been submitted to server but no redirect actions have run yet. Note that if directive callbacks are present no redirect action will be executed even if it is set on the form.

Global callbacks have a timeout of 5 seconds, after which submission text will be displayed and FF will try to execute redirect actions if any are present. Your callback at that point may still be running but it will do so in a way not obvious for the user; if the user chooses to navigate away from the page or redirect happens automatically, your long running callbacks will not complete. So make sure your callbacks don't take too long to execute.

How to setup a global callback

Set up is very simple, just follow these steps:

1. Attach a listener to the document and listen for "ffFormReady" event. The event contains **formInfo** object with form name, registration and completion notification functions.
2. Call **window**.**ffCallbacks**.*registerCallback* function with the following parameters: **formInfo** object, your function reference and execution context (undefined if your function is global). In your function you can expect **data** object and **formInfo** object. You MUST call **formInfo**.*notifyFormFactoryOfCompletion* function passing in your function's name, which must be unique, once your callback is finished processing. The **data** object contains action data and form fields (as key value pairs).

Here's a look at the **data** object:

```
data = {
        actionData : [],
        resultData : {
                textInput_0_1 : "Some value"
                ...
        }
}
```

Here's a look at the **formInfo** object:

```
formInfo = {
  formName: String,
  notifyFormFactoryOfCompletion: function,
  registerCallback: function
}
```

Here's how a sample callback setup looks:

```
<script type="text/javascript">
  document.addEventListener("ffFormReady", function(e) {
  var formInfo = e.formInfo;
  window.ffCallbacks.registerCallback(formInfo, testObject.testFunction, testObject);
});

var testObject = {
  testFunction : function(data, formInfo) {
        //Time out is just for example purposes to simulate an ajax request!!!
      setTimeout(function() {
          formInfo.notifyFormFactoryOfCompletion("testFunction");
      }, 3000);
  }
}
</script>
```

As you can see registering a global callback is very simple.

Directive callbacks

Directive callbacks allow you to remove the default submission message and insert your own directive in its place. It is totally up to you to define the behaviour of that directive. Once you have a directive callback there are two options: you can show or hide its template (for most configurations you will want to show the template). To configure this option go to metadata tab -> callbacks and toggle the **Display templates** switch as desired.

Example: Redirect timer callback

Creating a directive callback is easy, however, you must be comfortable with Angular and have a good understanding of directives. You must be using a module with a dependency on Form Factory. For an example of such a module you can see our form-factory-snippets-extension module. Once your dependencies are set up you can begin creating your callback by following the process documented below.

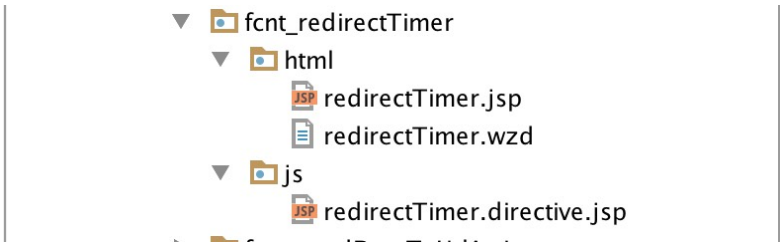
Definition

The first thing you need to do is to create a definition. The definition must inherit from *fcmix:callback* mixin. Here's what you will want to use most of the time:

```
[fcnt:redirectTimer] > jnt:content, fcmix:callback, mix:title, jmix:droppableContent, jmix:hiddenType
```

Folders and files

Next, you need to create view folder and view files. Create an **fcnt_redirectTimer** folder with **html** and **js** folders inside. Inside **html** folder create *redirectTimer.jsp* and *redirectTimer.wzd* files. Inside **js** folder create *redirectTimer.directive.jsp* file. You should get the following structure:



Wizard (redirectTimer.wzd)

Wizards for callbacks are very simple. The only property that you need to specify is the name of the callback. Note that the name is used to compile directive so it must be consistent with your directive name when it is lowercased and dasherized. Here's the wzd file for redirect timer callback:

```
callback {
  label "Redirect Timer"
}
```

Once again, according to the label, your callback directive will be compiled using "redirect-timer" tag.

Callback view (redirectTimer.jsp)

In this file you want to specify how your callback will look once compiled. Remember that this view is a template within the scope of your directive. For the timer, I decided to show a countdown and an optional link if the user doesn't want to wait and wishes to proceed somewhere else.

```
<div class="row">
  <div class="col-md-12">
      Redirecting to  <a href="{{page}}">{{page}}</a> in {{secondsToRedirect}} second(s).
  </div>
</div>
<div class="row">
  <div class="col-md-12">
      Feel free to look for more information at <a href="{{url}}">{{url}}</a>
  </div>
</div>
```

Create directive (redirectTimer.directive.jsp)

The directive file contains angular directive for the callback. Once again, it should be named according to the label you create, in this case it's "redirectTimer" or "redirect-timer" as it will be referred to on the DOM. The directive accepts **actionData** parameter, which is an array of action data objects returned by redirect actions. Let's take a look at the directive for the callback.

```
<%@ page contentType="text/javascript" %>
<%@ taglib prefix="formfactory" uri="http://www.jahia.org/formfactory/functions" %>
<%--@elvariable id="renderContext" type="org.jahia.services.render.RenderContext"--%>

  angular
      .module('formFactory')
      .directive('redirectTimer', ['$log', 'ffTemplateResolver', '$interval', function ($log, ffTemplateResolver,
$interval) {
          var directive = {
              restrict: 'E',
              require: ['^ffController'],
              scope: {
```

```
                actionData: '=',
                callback: '&'
            },
            templateUrl: function(el, attrs) {
                return ffTemplateResolver.resolveTemplatePath('${formfactory:addFormFactoryModulePath('/form-
factory-callbacks/redirect-timer', renderContext)}', attrs.viewType);
            },
            link: linkFunction
        };

        return directive;

        function linkFunction(scope, el, attr, ctrl) {
            console.log("redirect timer");
            scope.url = '#';
            scope.page = '#';
            scope.secondsToRedirect = 5;

            for (var i in scope.actionData) {
                if ('actionName' in scope.actionData[i]
                      && scope.actionData[i].actionName[0] === 'redirectToUrl') {
                    scope.url = scope.actionData[i].redirectUrl[0];
                }
                if ('actionName' in scope.actionData[i]
                      && scope.actionData[i].actionName[0] === 'redirectToAPage') {
                    scope.page = scope.actionData[i].redirectUrl[0];
                }
            }

            scope.countDown = function() {
                if (scope.secondsToRedirect === 0) {
                    scope.$emit("callbackDone", "redirect-timer");
                    window.location.assign(scope.page);
                    return;
                }
                scope.secondsToRedirect -= 1;
            };

            $interval(scope.countDown, 1000);
        }
    }]);
```

As you can see, you must use **ffTemplateResolver** to resolve callback's template. Callback templates are stored under "/form-factory-callbacks/" subpath and referenced by lowercased, dasherized label that you specify in the wzd file. In this case the callback is rather simple and does not involve any complex logic/actions so I can pack all the functionality inside the link function. However, you can specify a controller if it is necessary.

Note that in order for this directive callback to work, you need to add two actions to your form: redirect to url and redirect to page as it relies on the data returned by the actions. To make it available for your form you must deploy your module, add it to the site and select the callback from the list of callbacks found in the metadata panel. You can have multiple callbacks just make sure that they don't contradict each other.