Form Factory Prefill Guide

The pre-fill feature provides the following functionality:

- Upon load of the form prefill specific fields with custom data
- Make the prefilled field read only once prefilled
- Make the prefilled field hidden once prefilled

In this document I will look at how one can create a simple prefill.

Create definition

First, you need to create definition. It should look like so:

```
[fcnt:userInfoPrefill] > jnt:content, fcmix:prefill, mix:title, jmix:droppableContent,
jmix:hiddenType
```

Create wizard

Once the definition is created you need to create wizard file. By convention wizard files are placed inside **html** folder of the view. So in this case the wizard file goes inside fcnt_userInfoPrefill/html/**userInfoPrefill.wzd**.

```
package fcnt_userInfoPrefill.html

prefill {
   label "User info prefill"
   type "super"
   wizard "<ff-user-info view-type='designView'></ff-user-info>"
   serviceName "userInfoPrefillService"
   dataKey "name"
   availableDataKeys "name", "email", "path"

   properties {
       hideAfterPrefill "false"
       makeReadOnlyAfterPrefill "false"
   }
}
```

Let’s look at the wizard in more detail.

label - Specifies the name of our prefill. It will be used to create a node in JCR with name “user-info-prefill”.
type - Specifies type of prefill
wizard - Specifies angular directive which will be used to for prefill configuration
serviceName - Specifies the name of the service which is used to get prefill data
dataKey - Specifies default dataKey which will be used to prefill input
availableDataKeys - Provides a list of all available dataKey for this prefill
hideAfterPrefill - If set to true it hides input once it is prefilled
makeReadOnlyAfterPrefill - Is set to true is makes the input read-only after prefill

As you go through this guide these concepts should become more clear.

Create service

Service is the backbone of a prefill; it supplies data used for prefilling. Service needs to retrieve data, cache it, for performance considerations and return a promise for that data. Usually data needs to be preprocessed before being returned. Keep in mind that the format of returned data must be:

```
{
      availableDataKey1 : “<data>”,
      availableDataKey2 : “<data>”
      ...
}
```

The “<data>” part, in turn, must be compatible with input value expected by Form Factory for that particular input.

Services are normally stored in the **js** folder of the view. So in this case fcnt_userInfoPrefill/js/**userInfoPrefill.service.jsp**

```
<%@ page contentType="text/javascript" %>

(function () {
   'use strict';

   var userInfoPrefillService = function (contextualData, $q, $http, $timeout) {

       var dataCache = null;
       var deferred = null;

       /**
```

```
         * MUST IMPLEMENT
         *
         * Gets data for prefill which must be an object:  { key: value, ... }
         * @returns {Promise}
         */
        this.getData = function() {
            if (deferred !== null) {
                return deferred;
            }

            deferred = $q.defer();

            if (dataCache !== null) {
                $timeout(function(resolve, reject) {
                    deferred.resolve(dataCache);
                }, 100);
                return deferred;
            }


            $http({
                //Entry point implemented in form-factory-core module as an example
                url: contextualData.context + '/modules/formfactory/live/userinfo/' +
contextualData.locale,
                method: 'GET'
            }).then(function(data) {
                console.log("User info prefill service", data);
                //Manipulate data as you see fit

                //Cache data to avoid future requests
                dataCache = data.data;
                deferred.resolve(dataCache);
            }, function(error) {
                deferred.reject(error);
            });
            return deferred;
        }
    };

    angular.module('formFactory')
            .service('userInfoPrefillService', ["contextualData",
                "$q", "$http", "$timeout", userInfoPrefillService]);
})();
```

Keep in mind that you can use any services used in Form Factory.

Create directive

Directive is used to configure prefill in builder mode and can be either very simple or very complex depending on your needs. All of its components are essentially the same as those needed for inputs, validations or logics. If you haven't looked at those make sure to read a guide on input creation to have a better understanding of how input extension works if Form Factory.

One thing that needs to be done specifically for prefill directives is you need to indicate prefill name inside the link function, which is identical to the wizard label lowercased and dasherized.

Directives are placed alongside services. In this case fcnt_userInfoPrefill/js/**userInfoPrefill.directive.jsp**

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

(function() {
   var userInfoPrefill = function(ffTemplateResolver) {
       return {
           restrict: 'E',
           templateUrl: function(el, attrs) {
               return
ffTemplateResolver.resolveTemplatePath('${formfactory:addFormFactoryModulePath('/form-factor
y-prefills/user-info-prefill', renderContext)}', attrs.viewType);
           },
           link:linkFunc
       };

       function linkFunc(scope, el, attr, ctrl) {
           scope.prefillName = 'user-info-prefill';
       }
   };
   angular
       .module('formFactory')
       .directive('ffUserInfo', ['ffTemplateResolver', userInfoPrefill]);
})();
```

Create design view

Form Factory provides a default design view for your prefills which allows you to select *dataKey* and toggle *hideAfterPrefill* and *makeReadeOnlyAfterPrefill* fields. To get that basic functionallity all you need to do is create a file fcnt_userInfoPrefill/html/**userInfoPrefill.designView.jsp** with the following content:

```
<ff-prefill-selection prefill-name="prefillName"></ff-prefill-selection>
```

Note that prefill name will come from the scope variable that you declared in your directive.

Create resource bundle

In your resource bundle simply add the following:

```
ff.label.prefill.user-info-prefill=User Info Prefill
```

Again, notice how label in wizard is related to the key in resource bundle.

You are done!

Compile the module, add it to your site and navigate to edit an input. You should see a “Prefill” tab and the prefill you just created. Note that only one prefill can be activated on an input at a time.