How to develop a Mood Rating Form Field

In this tutorial, we will follow the guideline for developing a plugin to develop our Mood Rating Form Field plugin. Please also refer to the very first tutorial How to develop a Bean Shell Hash Variable for more details steps.

1. What is the problem?

We would like to have a rating field with some smiley images which can be reuse for other form.

2. How to solve the problem?

We will develop a Form Field Element Plugin to render our mood rating field.

3. What is the input needed for your plugin?

To develop a Mood Rating Form Field plugin, we will need to provide some standard inputs for a Form Field element.

1. Field Id
2. Field Label
3. Validator
4. Readonly
5. Workflow Variable

4. What is the output and expected outcome of your plugin?

A form field shown selection of smiley images and its radio button.

5. Are there any resources/API that can be reused?

To develop the Mood Rating Form Field plugin, we can extends the Radio field in core product then replace its template and plugin properties options.

6. Prepare your development environment

We need to always have our Joget Workflow Source Code ready and builded by following this guideline.

The following tutorial is prepared with a Macbook Pro and the Joget Source Code is version 5.0.1. Please refer to the Guideline for developing a plugin article for other platform commands.

Let's say our folder directory is as follows.

```
- Home
  - joget
    - plugins
    - jw-community
      -5.0.1
```

The "plugins" directory is the folder we will create and store all our plugins and the "jw-community" directory is where the Joget Workflow Source code is stored.

Run the following command to create a maven project in "plugins" directory.

```
cd joget/plugins/
~/joget/jw-community/5.0.1/wflow-plugin-archetype/create-plugin.sh org.joget mood_rating 5.0.1
```

Then, the shell script will ask us to key in a version number for the plugin and ask us for a confirmation before it generates the maven project.

```
Define value for property 'version':  1.0-SNAPSHOT: : 5.0.0
[INFO] Using property: package = org.joget
Confirm properties configuration:
groupId: org.joget
artifactId: mood_rating
version: 5.0.0
package: org.joget
Y: : y
```

We should get a "BUILD SUCCESS" message shown in our terminal and a "mood_rating" folder created in the "plugins" folder.

Open the maven project with your favourite IDE. I will be using NetBeans.

7. Just code it!

a. Extending the abstract class of a plugin type

Create a "MoodRatingField" class under "org.joget" package. Then, extend the class with **org.joget.apps.form.lib.Radio** class. The **org.joget.apps.form.lib.Radio** class is an implementation of **org.joget.apps.form.model.Element** abstract class. Please refer to Form Field Element Plugin.

b. Implement all the abstract methods

As usual, we have to implement all the abstract methods. We will use the AppPluginUtil.getMessage method to support i18n and using constant variable MESSAGE_PATH for message resource bundle directory.

Implementation of all basic abstract methods

```
package org.joget;

import java.util.Map;
import org.joget.apps.app.service.AppPluginUtil;
import org.joget.apps.app.service.AppUtil;
import org.joget.apps.form.lib.Radio;
import org.joget.apps.form.model.FormBuilderPalette;

public class MoodRatingField extends Radio {

    private final static String MESSAGE_PATH = "message/form/MoodRatingField";

    @Override
    public String getName() {
        return "Mood Rating";
    }

    @Override
    public String getVersion() {
        return "5.0.0";
    }

    @Override
    public String getClassName() {
        return getClass().getName();
    }

    @Override
    public String getFormBuilderCategory() {
        return FormBuilderPalette.CATEGORY_CUSTOM;
    }

    @Override
    public String getLabel() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.MoodRatingField.pluginLabel", getClassName(), MESSAGE_PATH);
    }

    @Override
    public String getDescription() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.MoodRatingField.pluginDesc", getClassName(), MESSAGE_PATH);
    }

    @Override
    public String getPropertyOptions() {
        return AppUtil.readPluginResource(getClass().getName(), "/properties/form/moodRatingField.json", null,
true, MESSAGE_PATH);
    }

    @Override
    public String getFormBuilderTemplate() {
        return "<label class='label'>"+getLabel()+"</label>";
    }
}
```

Now, we have to create a UI for admin user to provide inputs for our plugin. In getPropertyOptions method, we already specify our Plugin Properties Options definition file is located at "/properties/form/moodRatingField.json". Let us create a directory "resources/properties/form" under "mood_rating/src/main" directory. After creating the directory, create a file named "moodRatingField.json" in the "properties" folder.

In the properties definition options file, we will need to provide options as below. Please note that we can use "@@message.key@@" syntax to support i18n in our properties options.

```
[{
    title : '@@form.moodRating.config@@',
    properties : [{
        name : 'id',
        label : '@@form.radio.id@@',
        type : 'textfield',
        required : 'True',
        regex_validation : '^[a-zA-Z0-9_]+$',
        validation_message : '@@form.radio.invalidId@@'
    },
    {
        name : 'label',
        label : '@@form.radio.label@@',
        type : 'textfield',
        value : '@@org.joget.MoodRatingField.pluginLabel@@'
    }]
},
{
    title : '@@form.radio.advancedOptions@@',
    properties : [{
        label : '@@form.radio.data@@',
        type : 'header'
    },
    {
        name : 'validator',
        label : '@@form.radio.validator@@',
        type : 'elementselect',
        options_ajax : '[CONTEXT_PATH]/web/property/json/getElements?classname=org.joget.apps.form.model.
FormValidator',
        url : '[CONTEXT_PATH]/web/property/json[APP_PATH]/getPropertyOptions'
    },
    {
        label : '@@form.radio.ui@@',
        type : 'header'
    },
    {
        name : 'readonly',
        label : '@@form.radio.readonly@@',
        type : 'checkbox',
        value : 'False',
        options : [{
            value : 'true',
            label : ''
        }]
    },
    {
        label : '@@form.radio.workflow@@',
        type : 'header'
    },
    {
        name : 'workflowVariable',
        label : '@@form.radio.workflowVariable@@',
        type : 'textfield'
    }]
}]
```

After completing the properties option to collect input, we can work on the main methods of the plugin which are renderTemplate and formatData method. Since we extends Radio class, we do not need to implement formatData method.

```
    @Override
    public String renderTemplate(FormData formData, Map dataModel) {
        String template = "moodRatingField.ftl";

        // set value
        String value = FormUtil.getElementPropertyValue(this, formData);
        dataModel.put("value", value);
        String html = FormUtil.generateElementHtml(this, formData, template, dataModel);
        return html;
    }
```

In the renderTemplate, we specify the template file to "moodRatingField.ftl". Let create this file under "mood_rating/src/main/resources/templates" directory. Then, using FreeMaker syntax to construct our template as below:

```
<div class="form-cell mood_rating" ${elementMetaData!}>
    <label class="label">${element.properties.label} <span class="form-cell-validator">${decoration}</span><#if
error??> <span class="form-error-message">${error}</span></#if></label>
    <div class="form-cell-value" id="${elementParamName!}${element.properties.elementUniqueKey!}">
    <#if !(request.getAttribute("org.joget.MoodRatingField")??) >
        <style>
            .mood_rating .tdstyle {text-align:center;width:20%;border:0px none transparent !important;}
        </style>
    </#if>
    <table style="width:150px">
        <tbody>
            <tr>
                <td class="tdstyle"><img height="25" width="25" src="${request.contextPath}/plugin/org.joget.
MoodRatingField/images/smiley5.png"></td>
                <td class="tdstyle"><img height="25" width="25" src="${request.contextPath}/plugin/org.joget.
MoodRatingField/images/smiley4.png"></td>
                <td class="tdstyle"><img height="25" width="25" src="${request.contextPath}/plugin/org.joget.
MoodRatingField/images/smiley3.png"></td>
                <td class="tdstyle"><img height="25" width="25" src="${request.contextPath}/plugin/org.joget.
MoodRatingField/images/smiley2.png"></td>
                <td class="tdstyle"><img height="25" width="25" src="${request.contextPath}/plugin/org.joget.
MoodRatingField/images/smiley1.png"></td>
            </tr>
            <tr>
                <#list ['5', '4', '3', '2', '1'] as i>
                    <td class="tdstyle">
                        <input grouping="${elementParamName!}" id="${elementParamName!}"
name="${elementParamName!}" type="radio" value="${i}" <#if error??>class="form-error-cell"</#if> <#if element.
properties.readonly! == 'true'> disabled</#if> <#if value?? && value == i>checked</#if> />
                    </td>
                </#list>
            </tr>
        </tbody>
    </table>
    </div>
    <div style="clear:both;"></div>
</div>
```

There are some smiley image files will be used by the template, let put those image files under "mood_rating/src/main/resources/resources/image" directory.

c. Manage the dependency libraries of your plugin

There are no additional library needed.

d. Make your plugin internationalization (i18n) ready

We are using i18n message key in getLabel and getDescription method. We will use i18n message key in our properties options definition as well. Then, we will need to create a message resource bundle properties file for our plugin.

Create a directory, "resources/message/form", under "mood_rating/src/main" directory. Then, create a "MoodRatingField.properties" file in the folder. In the properties file, add all the message keys and its label as below.

```
org.joget.MoodRatingField.pluginLabel=Mood Rating
org.joget.MoodRatingField.pluginDesc=Form Field for rating mood
form.moodRating.config=Edit Mood Rating
```

e. Register your plugin to the Felix Framework

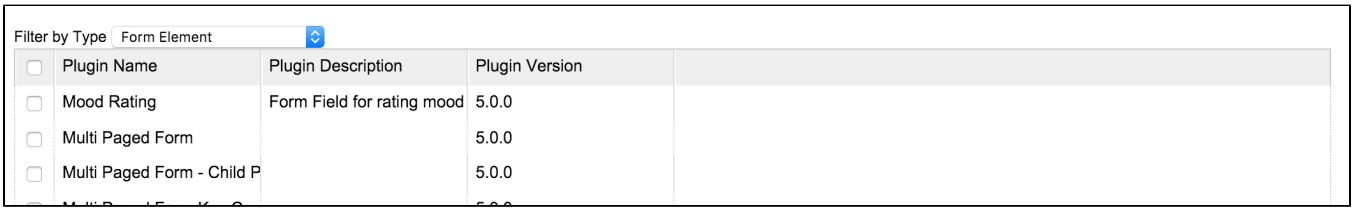
Next, we will have to register our plugin class in the Activator class (Auto generated in the same class package) to tell the Felix Framework that this is a plugin.

```
    public void start(BundleContext context) {
        registrationList = new ArrayList<ServiceRegistration>();
        //Register plugin here
        registrationList.add(context.registerService(MoodRatingField.class.getName(), new MoodRatingField(),
null));
    }
```

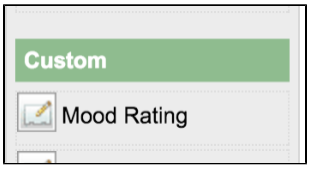
f. Build it and test

Let's build our plugin. Once the building process is done, we will find a "mood_rating-5.0.0.jar" file created under "mood_rating/target" directory.

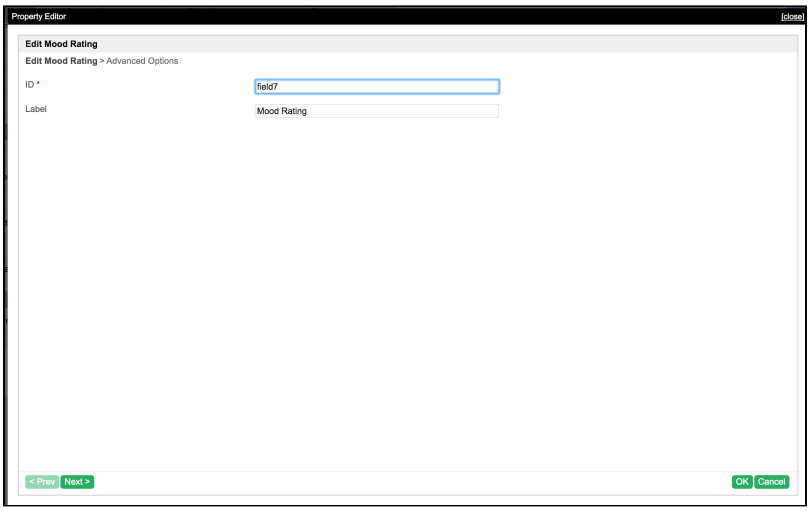
Then, let's upload the plugin jar to Manage Plugins. After uploading the jar file, double check that the plugin is uploaded and activated correctly.

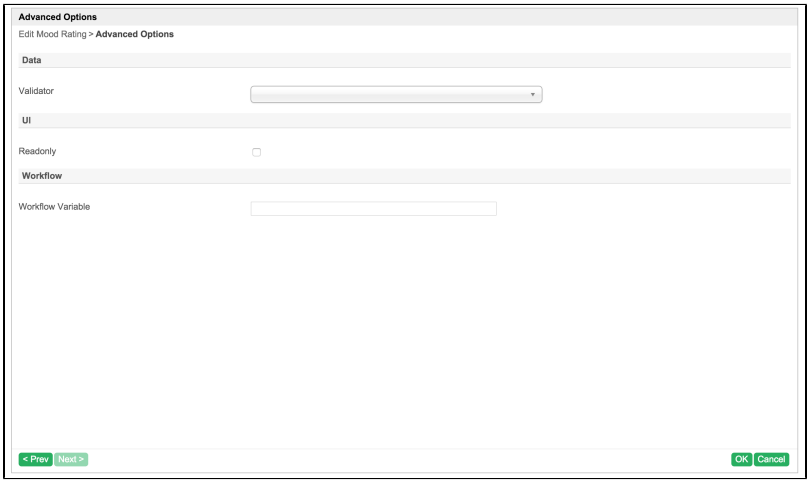


Then, check the Mood Rating field is shown in the Form Builder.

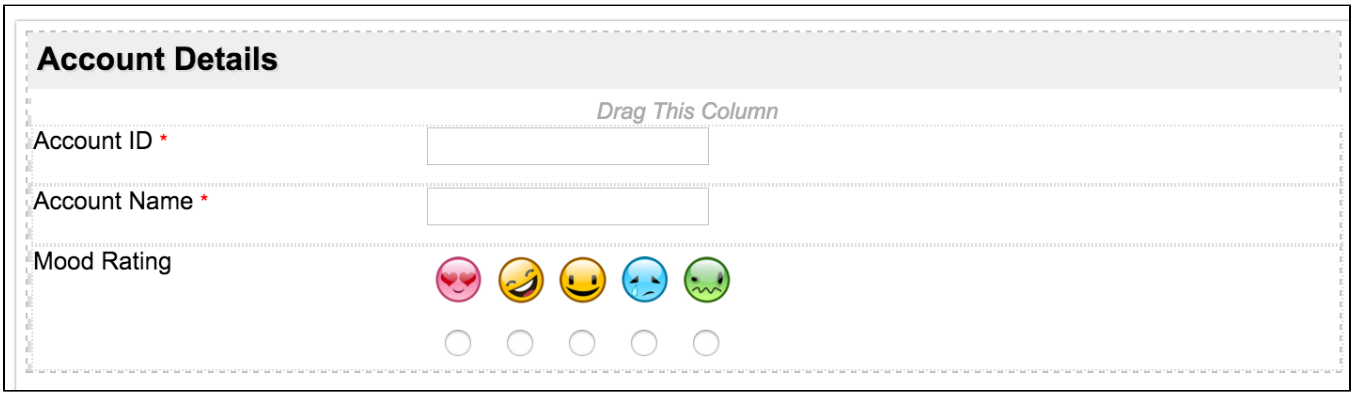


Drag it to the Form Builder Canvas and set its properties.

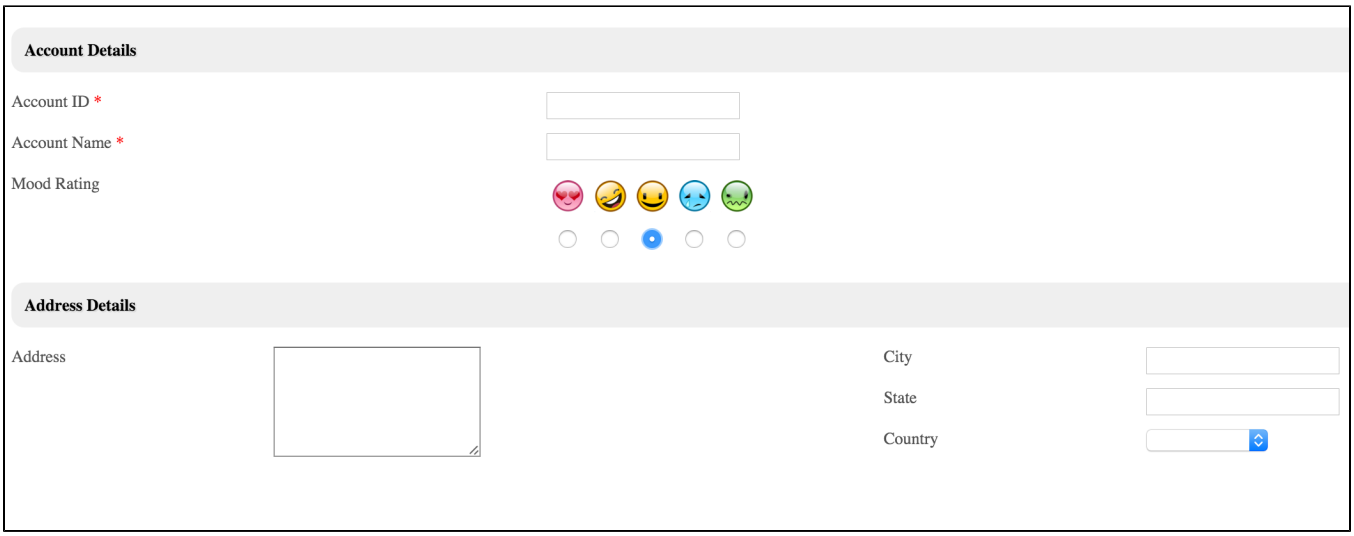




Save the properties and check the field is render in canvas as following.



Check and test the field in form.



8. Take a step further, share it or sell it

You can download the source code from mood_rating_src.zip.

To download the ready-to-use plugin jar, please find it in http://marketplace.joget.org/. (Coming Soon)