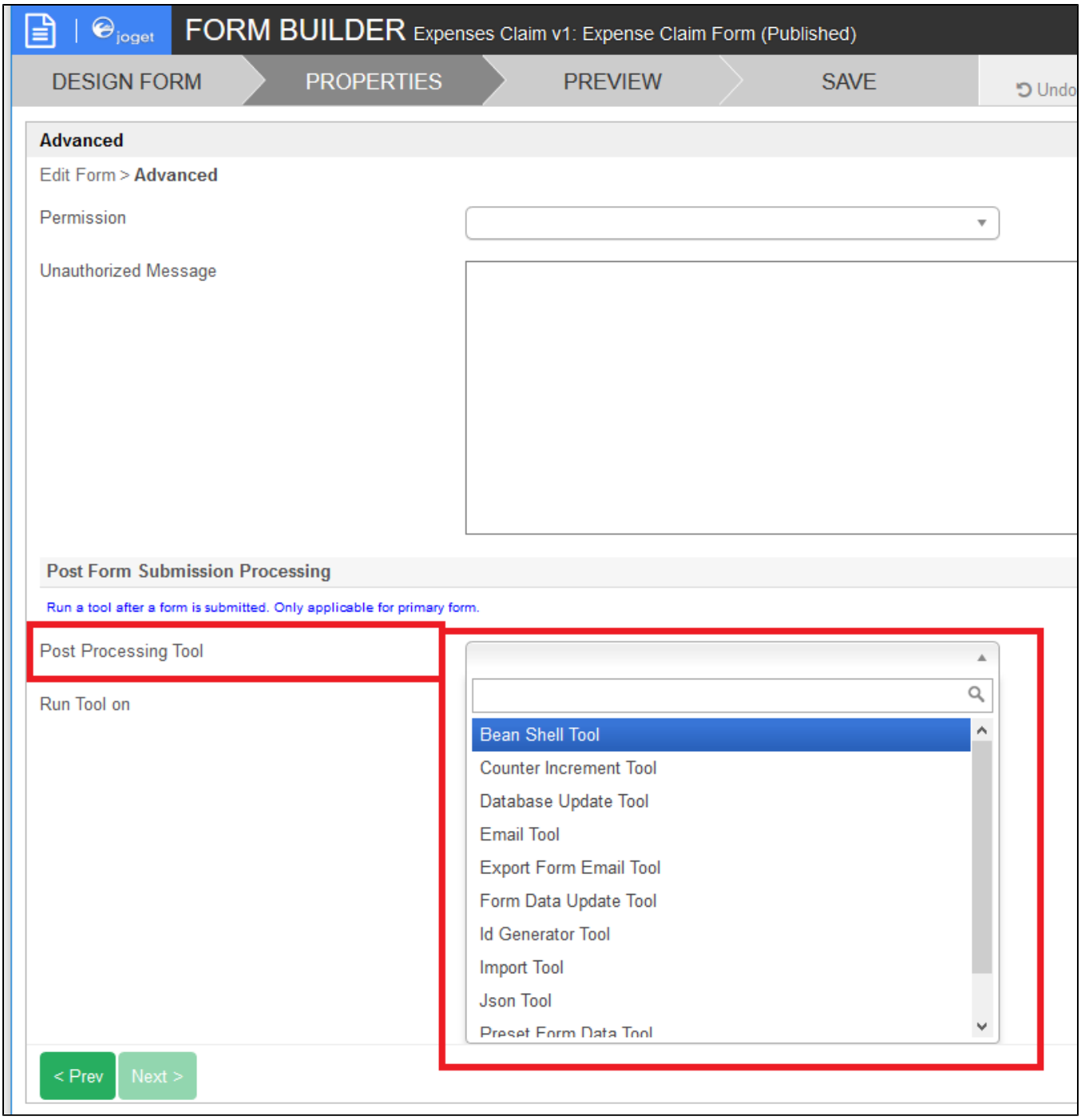
How to Include Existing Plugin Into Your Own Plugin

Take note of the "Post Processing Tool" attribute in a form builder's properties. We can see that it is listing down a list of Process Tool plugin for App Designer to choose.



Introducing the property definition type and configurations

We can trace this behavior to the property definition file "form.json" - https://github.com/jogetworkflow/jw-community/blob/6.0-SNAPSHOT/wflow-core/src/main/resources/properties/form/form.json#L81

This is the excerpt of the file that is responsible to generate such dropdown selection.

You will need to include such definition in your plugin's property definition file.

form.json

```
            {
                name:       'postProcessor',
                label:'@@form.form.postProcessor@@',
                type:'elementselect',
                options_ajax:'[CONTEXT_PATH]/web/property/json/getElements?classname=org.joget.plugin.base.
ApplicationPlugin',
                url:'[CONTEXT_PATH]/web/property/json[APP_PATH]/getPropertyOptions',
                default_property_values_url: '[CONTEXT_PATH]/web/property/json[APP_PATH]/getDefaultProperties'
            }
```

Explanations on the property definition code

Line 4 - Attribute "type" with the value set to "elementselect" is important to tell the property editor on how to handle the selection. A new tab will be created automatically upon a selection made.

Line 5 - Attribute "options_ajax" with the service URL set is responsible to load all the plugin tied to "org.joget.plugin.base.ApplicationPlugin" (Process Tool) into the selection. You may change the value to load the appropriate type of plugin you want.

[CONTEXT_PATH] is a special variable that will be replaced automatically at runtime with the Joget web app context information. (e.g. **http://localhost:8080/jw**)

Line 6 - With the value in Line 4 set, the property editor will make a call to this URL, passing along the selection made. (e.g. **http://localhost:8080/jw/web/property/json/expenseclaim/1/getPropertyOptions?value=org.joget.apps.app.lib.EmailTool**)

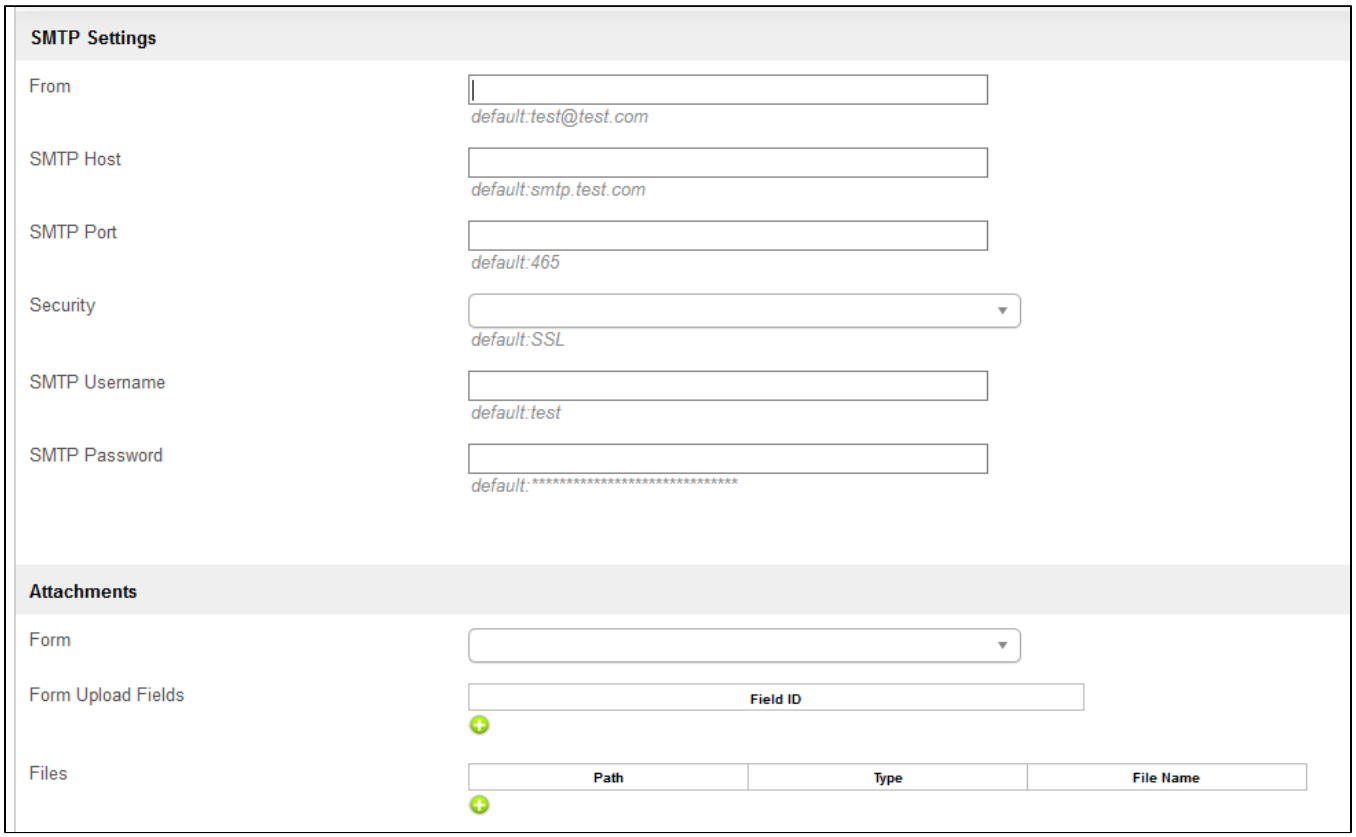
[CONTEXT_PATH] is a special variable that will be replaced automatically at runtime with the Joget webapp context information. (e.g. **http://localhost:8080/jw**)

[APP_PATH] is a special variable that will be replaced automatically at runtime with the Joget App context information. (e.g. **/expenseclaim/1**)

Line 7 - Attribute "default_property_values_url" is optional but recommended to be included. When a new tab is created for the configurations of the selected plugin, default values will be loaded to ease and shorten the time in configuring.

After a selection is made

After a selection is made, we can then see a new tab is added. We can also see that there are default values being loaded because of the inclusion of attribute "default_property_values_url" earlier.



How to retrieve the configurations set and run the plugin?

Using back the sample, we can refer to the code at https://github.com/jogetworkflow/jw-community/blob/6.0-SNAPSHOT/wflow-core/src/main/java/org/joget/apps/form/service/FormUtil.java#L1932. Refer to the code of the method "executePostFormSubmissionProccessor".

Essentially, we will just need these few lines in our own plugin to get it to work.

```
Object binderData = getProperty("postProcessor");
PluginManager pluginManager = (PluginManager) AppUtil.getApplicationContext().getBean("pluginManager");
if (binderData != null && binderData instanceof Map) {
        Map bdMap = (Map) binderData;
        if (bdMap != null && bdMap.containsKey("className") && !bdMap.get("className").toString().isEmpty()) {
                String className = bdMap.get("className").toString();
                Plugin p = pluginManager.getPlugin(className);

            Map propertiesMap = (Map) bdMap.get("properties");
                ApplicationPlugin appPlugin = (ApplicationPlugin) p;

                if (appPlugin instanceof PropertyEditable) {
                                ((PropertyEditable) appPlugin).setProperties(propertiesMap);
                }
                appPlugin.execute(propertiesMap);
        }
}
```

If we need to grab the plugin's default properties or to inject appDef, request object, etc then these excerpts from the method "executePostFormSubmissionProccessor" becomes necessary.

```
Map propertiesMap = null;

//get form json again to retrieve plugin properties
FormDefinitionDao formDefinitionDao = (FormDefinitionDao) FormUtil.getApplicationContext().getBean
("formDefinitionDao");
FormDefinition formDefinition = formDefinitionDao.loadById(form.getPropertyString(FormUtil.PROPERTY_ID),
appDef);
if (formDefinition != null) {
        String json = formDefinition.getJson();
        JSONObject obj = new JSONObject(json);
        JSONObject objProperty = obj.getJSONObject(FormUtil.PROPERTY_PROPERTIES);
        if (objProperty.has(FormUtil.PROPERTY_POST_PROCESSOR)) {
                JSONObject objProcessor = objProperty.getJSONObject(FormUtil.PROPERTY_POST_PROCESSOR);
                json = objProcessor.getString(FormUtil.PROPERTY_PROPERTIES);
                propertiesMap = AppPluginUtil.getDefaultProperties(p, json, appDef, ass);
        }
}
if (propertiesMap == null) {
        propertiesMap = AppPluginUtil.getDefaultProperties(p, (Map) temp.get(FormUtil.PROPERTY_PROPERTIES),
appDef, ass);
}
if (ass != null) {
        propertiesMap.put("workflowAssignment", ass);
}
propertiesMap.put("recordId", formData.getPrimaryKeyValue());
propertiesMap.put("pluginManager", pluginManager);
propertiesMap.put("appDef", appDef);
// add HttpServletRequest into the property map
try {
        HttpServletRequest request = WorkflowUtil.getHttpServletRequest();
        if (request != null) {
                propertiesMap.put("request", request);
        }
} catch (Exception e) {
        // ignore if class is not found
}
```